Combining computer algebra systems with **S**ymbolic **C**omputation **S**oftware **C**omposability **P**rotocol

Alexander Konovalov

Centre for Interdisciplinary Research in Computational Algebra
University of St Andrews, Scotland

Third RISC/SCIEnce Training School in Symbolic Computation, Linz, 18 July 2008

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

Acknowledgements

Thanks to:

- SCIEnce and *SCIEnce*tists, and especially to Temur Kutsia and John McDermott

The subject of the talk

Infrastructure for distributed symbolic computation:

- Underlying protocol: SCSCP
- Encoding for mathematical objects: OpenMath standard
- SCSCP implementation in GAP
- You should expect similar things to appear in any SCSCP-compliant system
- I will speak more about providing SCSCP services (a kind of web services), and small-scale parallelization, while the subsequent talks by Abyd Al Zain will introduce middleware for symbolic grids

SCIENCE Symbolic Computation Infrastructure for Europe

Modern needs of symbolic computation community

- Efficient tools for combining different computational algebra systems to solve complex problems that require capabilities not available in any single system
- Web services client and server interfaces allowing deployment of computer algebra systems as Web services and local/remote calls of facilities of another system in easy and efficient way
- This may be used to combine several copies of the same system in a parallel computing context of various scales from multicore to grids

SCIENCE Symbolic Computation Infrastructure for Europe

Most common restrictions from our GAP experience

- 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

SCIENCE Symbolic Computation Infrastructure for Europe

Our main directions of work

- Software composability:
  - A programme of standards developments and implementations for symbolic computation software to use Web services and OpenMath technologies, allowing them to be efficiently composed to solve complex problems
- Symbolic computing on the Grid:
  - developing common standards and middleware to allow the production of Grid-enabled symbolic computation systems
  - constructing research prototypes supporting appropriate security, scheduling, and resource broking for complex symbolic computing applications on computational Grids

SCIENCE Symbolic Computation Infrastructure for Europe

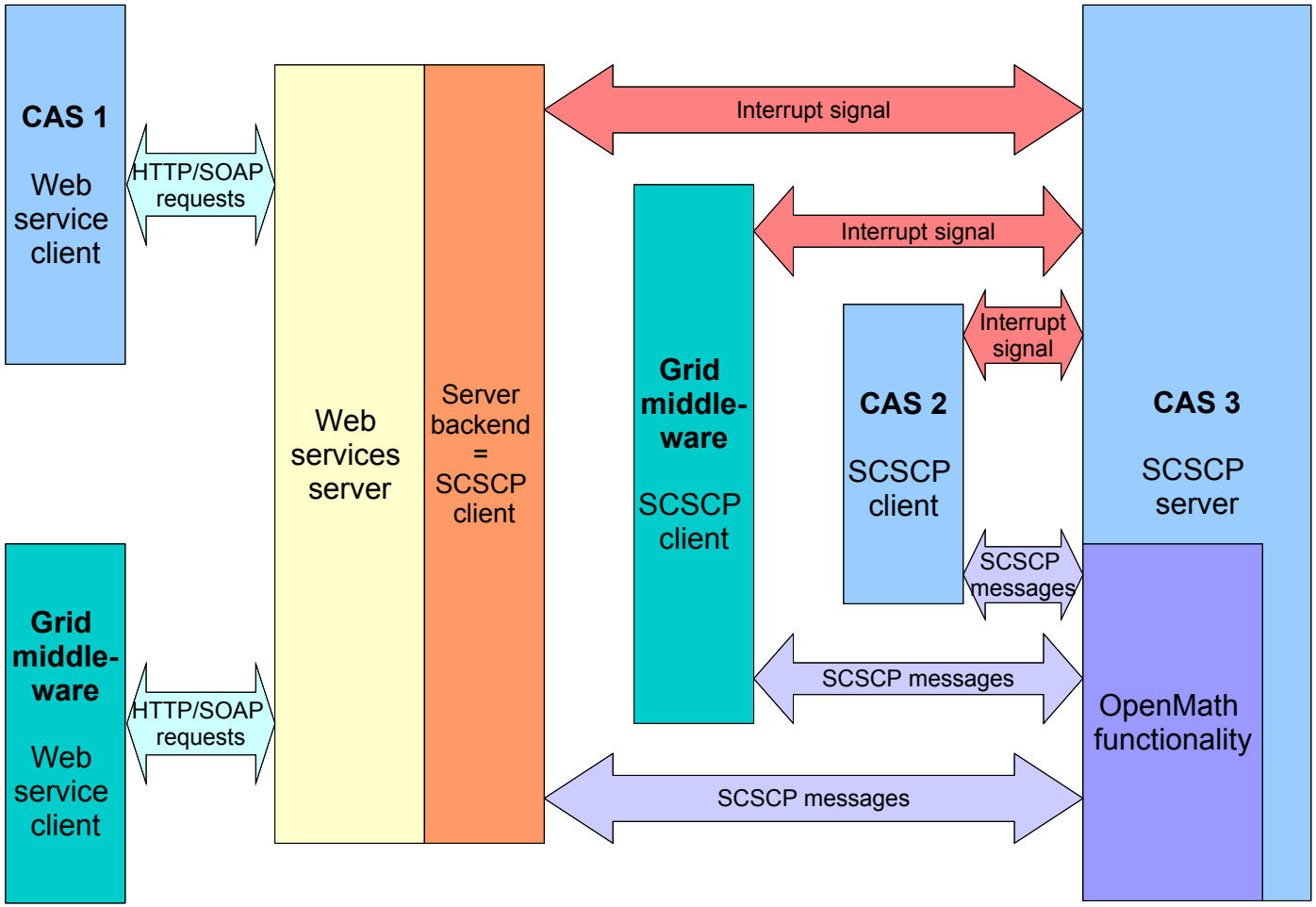
Common protocol for communication

In the direction of the software composability, on the first step we designed the *Symbolic Computation Software Composibility Protocol* (**SCSCP**) by which a computer algebra system (CAS) may offer services for the following clients:

- Another instance of the same CAS (in a parallel computing context or on different architectures)
- Another CAS running on the same computer or remotely
- A Web server which passes on the same services as Web services using SOAP/HTTP protocols to another clients
- Grid middleware

SCIENCE Symbolic Computation Infrastructure for Europe

Current vision of SCSCP usage



What is OpenMath?

- standard for representing mathematical objects with their semantics
- the current OpenMath Standard 2.0 is dated June 2004
- the worldwide OpenMath activities are coordinated within the OpenMath Society, based in Helsinki
- the idea is to allow their exchange between various programs, storing in databases, publishing on web . . .
- two encodings: XML and binary format

What is OpenMath?

- basic objects: integers, floats, strings, byte arrays, variables, symbols
- symbols consist of a name and a reference to a definition in an external document called *content dictionary* (**CD)**
- OpenMath objects can be combined recursively in a number of ways: application, attribution, binding, error
- support of various symbols may be different in various CASs, it may be complete (encoding and decoding) or only in one of these ways
- let us look at some examples in GAP
- see `http://www.openmath.org` for further details
- if you need support of existing symbols or interested in creating new CD - please contact us!

SCIENCE Symbolic Computation Infrastructure for Europe

SCSCP: OpenMath inside

- Protocol messages represented as OpenMath objects
- 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
- Both transmission of actual mathematical objects and references to them are supported
- Flexibility: service designer can choose the data to be `OMSTR`, `OMB`, `OMFOREIGN`, containing information in some other format, even including MathML encoding

SCIENCE Symbolic Computation Infrastructure for Europe

scscp1 CD defines:

- three main kinds of messages: `procedure_call`, `procedure_completed`, `procedure_terminated`
- options that may be added to the `procedure_call` message: `option_runtime`, `option_debuglevel`, `option_min_memory`, `option_max_memory`, `option_return_object`, `option_return_cookie`
- information that may be supplied with the result: `info_runtime`, `info_memory`
- standard errors: `error_runtime`, `error_memory`, `error_system_specific`

SCIENCE Symbolic Computation Infrastructure for Europe

scscp2 CD defines:

- procedures for remote objects: `store`, `retrieve`, `unbind`
- special procedures: `get_allowed_heads`, `get_transient_cd`, `get_signature`, `get_service_description`,
- special symbols: `signature`, `service_description`, `symbol_set`, `symbol_set_all`, `no_such_transient_cd`

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
- 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 4.4.10, which works in Linux, Windows and Mac OS
- Communication goes via TCP/IP protocol
- Handling OpenMath encoding, including:
  - new symbols from scscp1 and scscp2 OM CDs
  - support of OM attributes (OMATTR, OMATP)
  - support of OM references (OMR)

SCIENCE Symbolic Computation Infrastructure for Europe

User-level functionality:

- Installing procedures available as SCSCP services
- Running the SCSCP server
- Sending request to the server and getting result
- Store/Retrieve procedures allowing to work with remote objects
- InputOutputTCPStreams, compatible with other kinds of streams in GAP
- The underlying technology is well-hidden: we allow the end user to know nothing about OpenMath and SCSCP !!!

SCIENCE Symbolic Computation Infrastructure for Europe

Configuring SCSCP GAP server

1. Specify in `gap4r4/pkg/scscp/config.g`:
   - default InfoLevel, host name and port number
2. Put all what you need in the configuration file (see example in `gap4r4/pkg/scscp/tst/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( SCSCPserverAddress,
     SCSCPserverPort );
     ```
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 $\longrightarrow$ group id

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

list of permutations $\longrightarrow$ group id

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

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

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

Three approaches (continued)

group → group id

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

list of permutations → group id

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

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

pcgs code (integer that encodes the group) → 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

Before, we need an auxiliary function ...

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 ],
                          "chrystal.mcs.st-and.ac.uk",
                           26133 );
return result.object;
end;
```

Now we can see the demo!

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 $G = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$ be a permutation group. For the orbit computation, we need a service that takes an integer k and returns a set of all distinct images of k under $\sigma_1, \sigma_2, \ldots, \sigma_n$

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 see the demo, in which the first argument will be a remote object!

Example: remote objects

- `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)

SCIENCE Symbolic Computation Infrastructure for Europe

Parallelizing existing code with SCSCP is easy

- 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 version of GAP on multi-core machines
- Nevertheless, minimizing overhead might require some more tricks (optimizing data format, better scheduling, etc.)

SCIENCE Symbolic Computation Infrastructure for Europe

Example: communicating with other software

A Java program by Dan Roozemond searches in the On-Line Encyclopedia of Integer Sequences (`http://www.research.att.com/~njas/sequences/`)

What is the meaning of the following sequence:

1, 1, 1, 2, 1, 2, 1, 5, 2, 2, 1, 5, 1, 2, 1, 14 ?

SCIENCE Symbolic Computation Infrastructure for Europe

Example: communicating with other software

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	2	1	2	1	5	2	2	1	5	1	2	1	14

This is the number of groups of orders $1, 2, \ldots, 16$

Here we obtain a record with the search result:

```
gap> EvaluateBySCSCP("OnLineEncyclopediaOfIntegerSequences",
>    [ [ 1, 1, 1, 2, 1, 2, 1, 5, 2, 2, 1, 5, 1, 2, 1, 14 ] ],
>    "localhost", 26133 );
rec( object := [ 1, "A000001: Number of groups of order n." ],
attributes := [ [ "call_ID", "0" ] ] )
```

SCIENCE Symbolic Computation Infrastructure for Europe

*SCIEnce*tific work in progress

Next goals:

- Public releases of basic SCSCP implementations
- Extending OpenMath content dictionaries with more efficient encodings
- Adding support of selected OpenMath CDs to all participating systems
- Implementing higher level interfaces in all participating systems

Please let us know about computational services in which you might be interested !!!

SCIENCE Symbolic Computation Infrastructure for Europe

What should we have in the result

- low-overhead (compensated by easy-to-use), robust, cross-platforming, light-weight and reliable protocol
- possibility of communication not only between CASs but also between CAS and other software, including Web and Grid services
- besides the four participating systems (GAP, KANT, Maple and MuPAD), we expect more systems joining SCSCP framework later

References

SCIEnce — Symbolic Computation Infrastructure for Europe, project homepage
http://www.symbolic-computation.org/

S. Freundt, P. Horn, A. Konovalov, S. Linton, D. Roozemond. *Symbolic Computation Software Composability Protocol (SCSCP) specification*, Version 1.1, 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. Roozemond. *OpenMath Content Dictionary* **scscp1**
http://www.win.tue.nl/SCIEnce/cds/scscp1.html

D. Roozemond. *OpenMath Content Dictionary* **scscp2**
http://www.win.tue.nl/SCIEnce/cds/scscp2.html