SCSCP client and server for TRIP

M. Gastineau
gastineau@imcce.fr
Observatoire de Paris - IMCCE - CNRS
Astronomie et Systèmes Dynamiques
77, avenue Denfert Rochereau
75014 PARIS

© IMCCE - CNRS

September 17, 2012

Introduction

- TRIP supports the Symbolic Computation Software Composability Protocol (SCSCP) in accordance with the SCSCP specification version 1.3 and the OpenMath dictionaries scscp1 and scscp2.

- see http://www.symcomp.org/scscp, for more details related with SCSCP.

- Before using any functions about SCSCP in the TRIP system, you should execute the following command

source

```
include libscscpserver.t;
```

- It loads the standard SCSCP library for TRIP and it displays the following messages

output

```
> include libscscpserver.t;
Loading the SCSCP client/server library...
Registering the OpenMath CD...
>
```

SCSCP server

- To start the SCSCP server for the TRIP system, you should execute the macro scscp_runserver[**port**]. The default value for the port should be 26133.
- The global variables can be changed to modify the behavior of the SCSCP server. In the following example, we set the numerical mode to use arbitrary precision rational numbers instead of double precision floating-point numbers.

```
include libscscpserver.t;
_modenum=NUMRATMP;
%scscp_runserver[26133];
```

- It starts the scscp server in the TRIP session and it waits for any incoming connection on the port 26133.

```
> include libscscpserver.t;
Loading the SCSCP client/server library...
Registering the OpenMath CD...
> _modenum=NUMRATMP;
_modenum = NUMRATMP
> %scscp_runserver[26133];
Running the SCSCP server on port 26133...
Wait for connection on port 26133...
```

SCSCP server - provide additional SCSCP procedures

- The default content of $TRIPDIR/share/trip/lib/libscscpserver.t exports some basic SCSCP functions based on the standard Openmath Content Dictionnaries.

- Additional functions can be exported before running the SCSCP server. These functions added using the command scscp_map_macroassymbolcd defined in the libscscpserver.(so,dll). These functions must be defined as a TRIP macro.

- The following example exports the macro myscscp_evalmul as the symbol scscp_muleval of the CD SCSCP_transient_1

source

```
macro myscscp_evalmul[P1, P2, value]
 {
  t = value, value;
  q = evalnum(P1*P2, REAL, (x, t));
  return q[1];
 };
scscp_map_macroassymbolcd("myscscp_evalmul", "SCSCP_transient_1", "scscp_muleval");
```

SCSCP client connection

- TRIP provides functions to open and close a connection to a SCSCP server running locally or remotely.
  - ▶ scscp_connect : it starts a connection
  - ▶ scscp_close : it closes a connection
- This example displays the allowed symbols, to appear as the "head symbol", accepted by the SCSCP server located on this computer.

source

```
include libscscpserver.t;
fp = scscp_connect("localhost", 26133);
symboles = scscp_execute(fp,"object",
                         "scscp2",
                         "get_allowed_heads");
scscp_close(fp);
```

output

```
> include libscscpserver.t;
Loading the SCSCP client/server library...
Registering the OpenMath CD...
> fp = scscp_connect("localhost", 26133);
fp = SCSCP client connected to the SCSCP server
localhost
> symboles = scscp_execute(fp,"object",
"scscp2","get_allowed_heads");
symboles = openmath object '
<OMS cd="OMA" name="OMA"/>
<OMS cd="OM" name="OMF"/>
<OMS cd="OM" name="OMI"/>
<OMS cd="alg1" name="one"/>
<OMS cd="alg1" name="zero"/>
<OMS cd="arith1" name="abs"/>
<OMS cd="arith1" name="divide"/>
...
> scscp_close(fp);
```

Remote execution

- The SCSCP server can perform operations on the OpenMath objects or remote objects that are sent by the SCSCP client.

- The SCSCP client of TRIP performs these remote execution using the function scscp_execute. The result of this function depends on its second argument which specifies the type of the answer :
  - "nothing" : no value or reference is returned by the SCSCP server.
  - "cookie" : a reference to a remote object is returned by the server.
  - "object" : an OpenMath object is returned by the server.

- This example performs the operation $s = 5 \times P(x, y) \times Q(x, y)$ using the OpenMath symbol "times" of the CD "arith1".

```
include libscscpserver.t;
fp = scscp_connect("localhost", 26133);
P = 1+3*x^3+5*y^2$ Q = 1+3*x^2+5*y^4$
remoteP = scscp_put(fp,P)$
remoteQ = scscp_put(fp,Q)$
s = scscp_execute(fp,"object", "arith1", "times", 5, remoteP, remoteQ)$
scscp_close(fp);
```

Remote objects

The remote objects are the objects stored on the SCSCP server. The following operations can be performed on the remote objects.

- scscp_put : it sends a local value to the SCSCP server and store it as a remote object.This function returns a reference to the remote object.
- scscp_get : it retrieves the value of a remote object.
- scscp_delete : it deletes the remote object on the SCSCP server.

This example stores the polynomial $P(x, y) = 1 + 3x^3 + 5y^2$ on a remote SCSCP server. It performs the addition of $P(x, y) + (5 + y)$ and retrieve the result to q.

source

```
include libscscpserver.t;
fp = scscp_connect("localhost", 26133);
P = 1+3*x^3+5*y^2$
remoteP = scscp_put(fp,P);
remoteQ = scscp_execute(fp,"cookie",
                        "arith1","plus",
                        remoteP, 5+y)$
q = scscp_get(remoteQ);
scscp_delete(remoteP);
scscp_close(fp);
```

```
> include libscscpserver.t;
Loading the SCSCP client/server library...
Registering the OpenMath CD...
> fp = scscp_connect("localhost", 26133);
fp=SCSCP client connected to the SCSCP server localhost
> P = 1+3*x^3+5*y^2 $
> remoteP = scscp_put(fp,P);
remoteP = remote object "TempOID2@localhost:26133"
> q = scscp_get(remoteP);
q(x,y) =
1
+ 5*y**2
+ 3*x**3
> scscp_delete(remoteP);
> scscp_close(fp);
```

Disable unsupported content dictionnary

If the client or the server doesnot support a content dictionnary used by the TRIP for the OpenMath objects, the user could disable the content dictionnary used by TRIP for the communications.

- TRIP acts a SCSCP client. The command scscp_disable_cd allow to disable a content dictionnary when TRIP sends an OpenMath object to the server.

- TRIP acts a SCSCP server. The symbol serverdisablecd of the content dictionnary sscscp_transient_1 allow to disable a content dictionnary when TRIP sends an OpenMath object to the client.

- We run the Maple 16 system as the SCSCP client with its SCSCP package and TRIP acts as the SCSCP server. As maple 16 does not support the content dictionary polyr and and polyu, we disable it on the server side.

Maple output

```
> with(SCSCP):
> with(Client):
> cookie := StorePersistent("localhost:26133", x*y+1);
        cookie := "TempOID1@localhost:26133"

> Retrieve("localhost:26133", cookie);

Error, (in SCSCP:-Client:-ExtractAnswer) unsupported_CD, <OMS cd = 'polyr' name = 'term'/>

> CallService("localhost", "scscp\_transient\_1",
              "serverdisablecd", ["polyr"]):
> CallService("localhost", "scscp\_transient\_1",
              "serverdisablecd", ["polyu"]):
>  Retrieve("localhost:26133", cookie);
              1.0000000000000000 + 1.0000000000000000 x y
```

Example I (source)

- We run the GAP system as the SCSCP client with its SCSCP package and TRIP acts as the SCSCP server. We evaluate for $x = 5$ the product of two multivariate polynomials $p1(x)$ and $p2(x)$.

GAP source - SCSCP client

```
port:=26133;
x:=Indeterminate(Rationals,"x");
p1:=UnivariatePolynomial(Rationals,[3,2,3,4],x);
p2:=UnivariatePolynomial(Rationals,[1,0,0,2],x);
s1:=StoreAsRemoteObject( p1, "localhost", port );
s2:=StoreAsRemoteObject( p2, "localhost", port );
EvaluateBySCSCP( "scscp_muleval", [p1, p2, 5], "localhost", port );
```

TRIP source - SCSCP server

```
include libscscpserver.t;
_modenum=NUMRATMP;
macro myscscp_evalmul[P1, P2, value]
 {
  t=value,value;
  q=evalnum(P1*P2,REAL, (x,t));
  return q[1];
 };
scscp_map_macroassymbolcd("myscscp_evalmul", "SCSCP_transient_1","scscp_muleval");
%scscp_runserver[26133];
```

Example I (output)

GAP output

```
gap> port:=26133;
26133
gap> x:=Indeterminate(Rationals,"x");
x
gap> p1:=UnivariatePolynomial(Rationals,[3,2,3,4],x);
4*x^3+3*x^2+2*x+3
gap> p2:=UnivariatePolynomial(Rationals,[1,0,0,2],x);
2*x^3+1
gap> s1:=StoreAsRemoteObject( p1, "localhost", port );
#I  Creating a socket ...
#I  Connecting to a remote socket via TCP/IP ...
#I  Got connection initiation message
#I  Request sent ...
#I  Waiting for reply ...
< remote object TempOID1@localhost:26133 >
gap> s2:=StoreAsRemoteObject( p2, "localhost", port );
#I  Creating a socket ...
#I  Connecting to a remote socket via TCP/IP ...
#I  Got connection initiation message
#I  Request sent ...
#I  Waiting for reply ...
< remote object TempOID2@localhost:26133 >
gap> EvaluateBySCSCP( "scscp_muleval", [p1, p2,  5], "localhost", port );
#I  Creating a socket ...
#I  Connecting to a remote socket via TCP/IP ...
#I  Got connection initiation message
#I  Request sent ...
#I  Waiting for reply ...
rec( object := 147588, attributes := [ [ "call_ID", "0" ], [ "info_runtime", 0 ] ] )
gap>
```

Example II

- Now, TRIP is the SCSCP client and GAP runs as the SCSCP server. We export as SCSCP procedure "IdGroupDIN" the GAP function to compute for the dihedral group of order N its catalogue number in the GAP small groups library. Then we call this procedure to determine the catalogue number of the dihedral group of order 256.

GAP source - SCSCP server

```
LoadPackage("scscp");
IdGroupDIN:=function( n )
return IdGroup( DihedralGroup( n ) );
end;
InstallSCSCPprocedure("IdGroupDIN", IdGroupDIN );
ReadPackage("scscp/lib/errors.g");
RunSCSCPserver( "localhost", 26133 );
```

TRIP source - SCSCP client

```
include libscscpserver.t;
_modenum=NUMRATMP;
fp = scscp_connect("localhost",26133);
symboles = scscp_execute(fp, "object", "scscp2", "get_allowed_heads");
rescookie = scscp_execute(fp, "cookie", "SCSCP_transient_1", "IdGroupDIN", 256);
val = scscp_get(rescookie );
scscp_close(fp);
afftab(val);
```

Example II (output)

TRIP output

```
> include libscscpserver.t;
Loading the SCSCP client/server library...
Registering the OpenMath CD...
> _modenum=NUMRATMP;
                _modenum    =     NUMRATMP
> fp = scscp_connect("localhost",26133);
fp  = SCSCP client connected to the SCSCP server localhost
> symboles = scscp_execute(fp, "object",  "scscp2",  "get_allowed_heads");
symboles = openmath object '
<OMS cd="arith1" name="abs"/>

...

<OMS cd="scscp2" name="get_allowed_heads"/>
<OMS cd="scscp2" name="get_service_description"/>
<OMS cd="scscp2" name="get_transient_cd"/>
<OMS cd="scscp2" name="get_signature"/>
<OMS cd="meta" name="CDName"/>
<OMS cd="SCSCP_transient_1" name="IdGroupDIN"/>'
> rescookie = scscp_execute(fp, "cookie", "SCSCP_transient_1", "IdGroupDIN", 256);
rescookie  = remote object "TEMPVarSCSCP1@localhost:26133"
> val = scscp_get(rescookie );

val [1:2 ]       nb elements = 2

> scscp_close(fp);
> afftab(val);
val[1] = 256
val[2] = 539
```

For Further Reading

M. Gastineau, J. Laskar.

TRIP 1.2, Manuel de référence de TRIP, 2012.
IMCCE, Paris Observatory, http://www.imcce.fr/trip/.

S.Freundt, P.Horn, A.Konovalov, S.Linton, D.Roozemond.

Symbolic Computation Software Composability Protocol (SCSCP) specification, Version 1.3, 2009.
http://www.symcomp.org/scscp.

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.

A. Konovalov, S. Linton

SCSCP - Symbolic Computation Software Composability Protocol, GAP package, in development, 2008
http://www.cs.st-andrews.ac.uk/~alexk/scscp/.