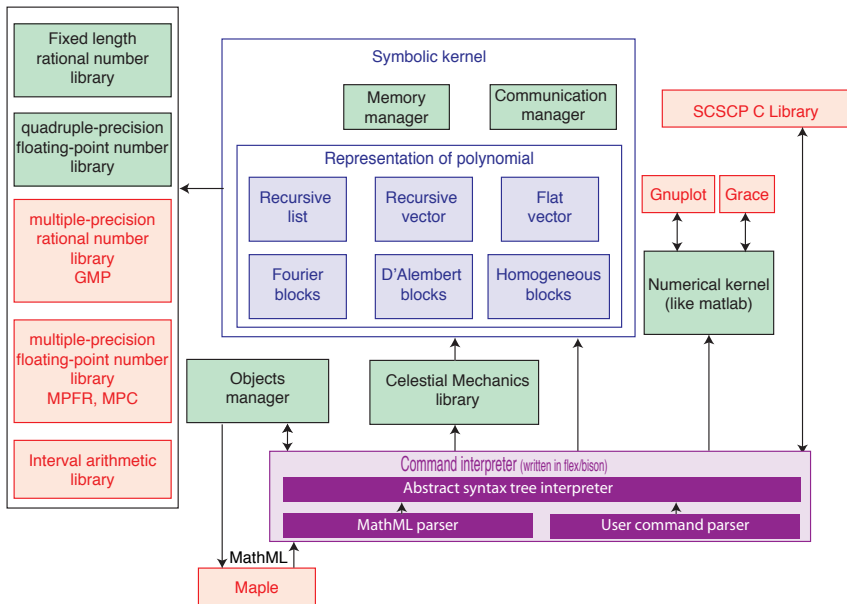


- TRIP is a general computer algebra system dedicated to celestial mechanics (J. Laskar, 1988-2009, M. Gastineau, 1998-2009)
- Programming language and interpreter.
loops, macros, conditions, ...
- Multivariate generalized formal series.
Use of angular variables
- parallelized kernel on SMP.
- usual operators (+,., inversion), Substitution, derivation, integration, fast evaluation
- Automatic or manual management for the variable order.
- Truncatures.
- Celestial mechanics functions: Poisson bracket, r/a , $\cos E$, ...
- Interface to dynamic library (DLL), such as lapack,

TRIP - architecture



- Sparse Polynomial Division Using a Heap (Monagan, Pearce, 2008)

Let $f = (1 + x + y^2 + z^3 + t^5 + u^7)^{12}$ and $g = (1 + u + t^2 + z^3 + y^5 + x^7)^{12}$.

Table 5. Very sparse multiplication and division over \mathbb{Z}

6188 × 6188 = 13209653 terms	$p = f \cdot g$	$q = p/f$
sdmp (1 word monomial)	2.12 s (202.2 MB)	2.60 s (1.0 MB)
sdmp (5 word monomial)	5.74 s (606.6 MB)	7.41 s (2.0 MB)
TriP v0.99 (floating point)	2.54 s (648.4 MB)	-
TriP v0.99 (rationals)	5.63 s (1255.1 MB)	-
Pari/GP 2.3.3	284.19 s	134.87 s
Magma V2.14-7	36.30 s (4155.5 MB)	276.45 s (405.2 MB)
Singular 3-0-4	44.00 s (1197.2 MB)	84.00 s (1013.3 MB)
Maple 12	250.24 s (2350.2 MB)	277.65 s (1522.0 MB)

TRIP - integration of SCSCP

- Allows TRIP to work as an SCSCP server and client on every supported platform (Linux, Mac OS X, Windows, OpenBSD)
- Integration in TRIP kernel
 - Handle the OpenMath objects and encodings (OMR, ...)
 - Add OpenMath content dictionaries management
 - Forward errors to the SCSCP server
- User-level facilities
 - Register new Openmath Content Dictionnaires and symbols
 - Provide new services to the SCSCP server
 - Work with remote objects
 - Perform remote execution

- To start the SCSCP server for the TRIP system, users execute the macro `scscp_runserver[port]`.
- Execute any command before the SCSCP server to modify the behavior of the server.

source

```
include libscscpserver.t; /* register openmath CD (e.g., polyu) */  
_modenum=NUMQUAD; /*switch quadruple-precision */  
%scscp_runserver [26133]; /* start SCSCP server */
```

output

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

TRIP - additional SCSCP procedures

- Additional functions can be exported before running the SCSCP server.
- These functions are added using the command `scscp_map_macroassymbolcd` defined in the `libscscpserver.(so,dll)`. These functions must be defined as a TRIP macro.
- Example : export the macro `myscscp_evalmul` as the symbol `scscp_muleval` of the CD `SCSCP_transient_1`

source

```
/*define a subroutine to evaluate P1(x).P2(x) at some value. */
macro myscscp_evalmul[P1, P2, value]
{
  t = value, value;
  q = evalnum(P1*P2,REAL, (x,t));
  return q[1];
};

/*register this subroutine as a symbol */
scscp_map_macroassymbolcd("myscscp_evalmul", "SCSCP_transient_1", "scscp_muleval");
```

Example 1 (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 $p_1(x)$ and $p_2(x)$.

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 1 (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>
```

TRIP - SCSCP client

- TRIP provides functions to open and close a connection to a SCSCP server running locally or remotely.
 - `object = scscp_connect(hostname, port)` : it starts a connection
 - `scscp_close(object)` : it closes a connection
- Example : Retrieve the list of procedures, to appear as the "head symbol", accepted by the SCSCP server.

source

```
include libscscpserver.t;  
  
/* create connection to SCSCP server */  
fp = scscp_connect("localhost", 26133);  
  
/* find the list of procedures supported  
by the SCSCP server */  
execsymbols = scscp_execute(fp, "object",  
"scscp2",  
"get_allowed_heads");  
  
/* close the connection */  
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="transc1" name="arccos"/>  
<OMS cd="transc1" name="arccosh"/>  
<OMS cd="alg1" name="one"/>  
<OMS cd="alg1" name="zero"/>  
<OMS cd="arith1" name="abs"/>  
...'  
> scscp_close(fp);
```

- SCSCP client performs these remote procedure call using the function `scscp_execute`(*client, answer, cd, symbol, ...*)
- Type of the answer
 - "nothing" (⇔ `option_return_nothing`) : no value or reference.
 - "cookie" (⇔ `option_return_cookie`) : a reference to a remote object.
 - "object" (⇔ `option_return_object`) : an OpenMath object.
- Example : $Q = 5 \times (1 + 3 * x^3 + 5 * y^2) \times (1 + x + y + z + t + u)^{12}$.

source

```
include libscscpserver.t;
fp = scscp_connect("localhost", 26133);
remoteS = scscp_execute(fp,"cookie", "arith1", "power",1+x+y+z+t+u,12)$
Q = scscp_execute(fp,"object", "arith1", "times", 5, 1+3*x^3+5*y^2, remoteS)$
scscp_close(fp);
```

TRIP - Remote objects

Client handles remote objects on the server

- **scscp_put** (\Leftrightarrow store, scscp2) : it sends a local value to the SCSCP server and store it as a remote object.
- **scscp_get** (\Leftrightarrow retrieve, scscp2) : it retrieves the value of a remote object.
- **scscp_delete** (\Leftrightarrow unbind, scscp2) : it deletes the remote object.

Client

```
> include libscscpserver.t;
Loading the SCSCP client/server library...
Registering the OpenMath CD...
> fp = scscp_connect("ptolemea.imcce.fr", 26134);
fp = SCSCP client connected to the SCSCP server
ptolemea.imcce.fr
> P = 1+3*x^3+5*y^2 $
> remoteP = scscp_put(fp,P);
remoteP = remote object "TempOID1@localhost:26134"
> q = scscp_get(remoteP);
q(x,y) =
1
+ 5*y**2
+ 3*x**3
> scscp_delete(remoteP);
> scscp_close(fp);
```

Server

```
> include libscscpserver.t;
Loading the SCSCP client/server library...
Registering the OpenMath CD...
> %scscp_runserver[26134];
Running the SCSCP server on port 26134...

Wait for connection on port 26134...
Connection received on port 26134
Receive a valid command
TempOID1(x,y) = 1 + 5*y**2 + 3*x**3
Send the answer
Receive a valid command
Send the answer
...
Wait for connection on port 26134...
```

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
```