Summer 2017 Internship Report

Victor Vasilyev

September 1, 2017

1 Introduction

During this summer, I undertook an internship to complete some tasks set by the OpenDreamKit project. The focus of the internship was the Maths-in-the-Middle paradigm (MitM)- an idea of establishing a universal communication system between different computer algebra systems (CAS) that would allow researchers to exploit their best features while solving complex problems. The tasks were:

1. Define group theory, as it is defined and implemented in the GAP computer algebra system, in the MMT language so that it can later be used for automation of alignment between a GAP SCSCP server and the MitM server.

2. Create an MitM proof-of-concept study that would showcase a problem being solved using miltiple CAS queried through an MitM server without the user having any knowledge of the underlying systems.

2 Definition of GAP Type System in MMT

In order to ensure smooth interaction between different algebra systems, the MMT server must have access to formalized concepts represented by them. The first step was thus to formalize the concepts of group theory available in the GAP system in the MMT language. The resulting code can be seen in MitM/groups repository on Mathhub[6].

3 Proof of MitM Concept

3.1 MitM Protocol

For modular interaction between a Maths-in-the-Middle server, CAS SCSCP servers and naive clients, I have developed the MitM/SCSCP protocol that allows CAS servers to go online and offline independently of the MitM server. Below is the specification of the protocol.

3.1.1 Peers

One of the systems in the dialog must initially act as a server. This server will be called the MitM server and expose a strict set of function headers to clients upon startup. The other system will be referred to from now on as CAS (Computer Algebra System) and can expose an arbitrary set of function headers.

3.1.2 Interaction

The MitM server must, upon boot-up, expose the symbol "eq" from the CD "relation", as well as the following symbols from the CD "mitm_transient" over SCSCP:

- **registerServer**, taking one compulsory argument, the address of the CAS server as a string, and one optional argument, the port at which the CAS serves SCSCP. If a port is not provided, it should default to 26133. This function should establish an SCSCP connection with the CAS server at that address and return the name/handler that the MitM server assigns to the CAS instance as a string. This name is used when aligning functions.

- **registerFunction**, taking three arguments: the handler of the server that exposes a symbol, the symbol that the server exposes, and a symbol in a global CD that it corresponds to. After this call, the MitM server should expose the global symbol as a function header and redirect calls to it to the CAS.

- **getAllServers**, taking no arguments. This function should return the list of handlers for all the servers MitM is currently aware of.

- **removeFunction**, taking two arguments, the handler of a CAS server and a symbol that that CAS exposes. After the call to this function no requests should be redirected to that symbol on that CAS.

- **removeServer**, taking one argument- the name/handler that refers to the client connected to the CAS. This function should close the SCSCP client to that CAS and stop serving functions that depend on symbols exposed by that CAS.

- **registerEquality**, taking three arguments: the handler of the server that can compare applications of an OpenMath symbol, the function header exposed by that server that compares the applications, and the symbol that the server can compare. After this function call, any calls to the "eq" symbol with the applications of this symbol as arguments should be redirected to the CAS.

3.2 Overview of the System

In order to apply the MitM paradigm to practice, I have created a case study[8] in which a user is able to access the functionality of two very different computer algebra systems by querying the MitM server that is implemented by an extension in MMT[7]. The study consists of five components:

- MitM server (as an MMT extension)
- GAP SCSCP server that specializes on computations in group theory
- Singular SCSCP server that specializes on polynomial calculations
- Python client that uses the SCSCP package
- Python script that lines up the interactions between the MitM server and CAS servers

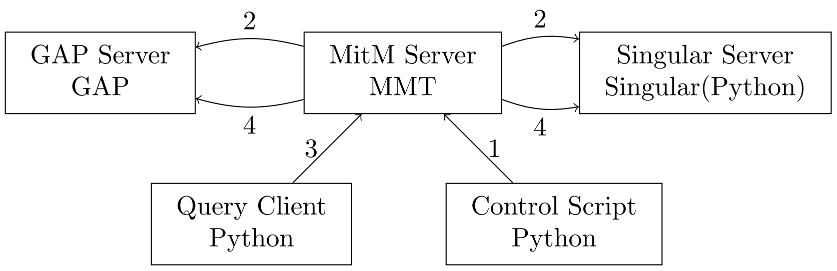

Figure 1: System Designed to Prove the Concept of the MitM Protocol

The procedure illustrated in figure 1 is as follows:

1. The control script establishes an SCSCP connection with the MMT MitM server and aligns the functions exposed by the CAS (GAP and Singular) with global symbols.
2. The MitM server establishes SCSCP connections with the CAS server at provided addresses.
3. The query client queries the MitM server for symbols from public CDs without any knowledge of the CAS in operation to calculate orbits of polynomials.
4. MitM server redirects the requests from the query client to the necessary CAS.

3.3 Components of the Study System

3.3.1 MMT MitM Server

The MitM server is implemented as an MMT-shell extension. The integration into the MMT ecosystem will be useful in the future for the automation of function alignment and type-checking of arguments. Currently only minimal arguments type-checking is done, to the extent that the server is able to check whether the arguments passed are strings, integers or symbols in OpenMath.

3.3.2 GAP SCSCP Server

GAP server uses the GAP SCSCP package to expose the minimal functions needed for this study- the constructor for a symmetric group of size N, and a function that takes a list and a symmetric group and returns a permutation of the list for every member of the symmetric group that is obtained by applying that member to every item of the list.

3.3.3 Singular SCSCP Server

Since Singular currently doesn't have any support for SCSCP, the Singular SCSCP server is written in Python using the scscp[4] and PySingular[5] python modules and is adapted from the example code on the py-scscp repository. The only function exposed by this server is the evaluation of equality of two applications of the DMP symbol.

3.3.4 Controlling Client Script

In the long term, the function alignment will be done automatically when a CAS informs the MitM server of its presence. Currently, however, the export of CAS type systems is still a work in progress, so the function alignment is done via the MitM protocol. The job of the controlling script is thus to align the constructor of symmetric group of size N and the orbit of a list to their respective symbols in public CDs, as well as registering the ability of the Singular server to equate polynomials.

3.3.5 Naive Client

The main client that queries the MitM server has no knowledge of the underlying CAS. It follows the procedure:

1. Create an OpenMath polynomial.
2. Obtain a symmetric group of size that is equal to the number of variables in the polynomial from MitM.
3. Using the obtained group, query MitM for all permutations of the list of variables.

4. Create polynomials from the permutations of the list of variables.
5. Filter out the duplicate polynomials by querying MitM for equality of polynomials.

While this is very much a brute-force algorithm to calculate an orbit of a polynomial, it showcases the ability of the client to query the MitM server that is then forced to use multiple CAS without the client needing any knowledge of the underlying systems.

3.4 Dependencies

The dependencies of the software in subject of the study and instructions on acquiring them:

- **Jupyter**[2]
- **GAP**[1]
- **Singular**[3]
- **MMT**[7]
- **Py-SCSCP**[4]
- **PySingular**[5]

3.5 Execution

Instructions on running the system:

- Run singular_server.py script using python, gap_server.g script using GAP, and start MMT. From the MMT shell prompt, run "file mitm_server.msl".
- Run ControllingClient.py using python to set up alignments between MitM server and CAS servers.
- Start a Jupyter kernel using "jupyter notebook" from command line and open the QueryingClient.ipynb notebook. This is the example procedure that showcases a few examples of calculations of orbits of polynomials.

4 Conclusion

4.1 Evaluation of the Study

This study has implemented the MitM approach using SCSCP, showing that the MitM paradigm is an achievable goal. Currently, however, the MitM server acts as merely a proxy, redirecting SCSCP requests to CAS that know how to evaluate them. For MitM to be a truly modular abstract algebra environment, the following changes must be made:

- A peer-to-peer connection must be made with the CAS servers instead of a simple client-serve one, so that CAS servers can, in turn, query MitM if during a computation they encounter a concept that lies outside their field of knowledge. In application to this particular case, it would be cleaner if, instead of asking MitM to produce permutations of a list, the client simply queries MitM for the orbit of a polynomial by defining an action of a member of the symmetric group on a polynomial. GAP would then be able to calculate the orbit by making the group act on the polynomial with the described action and querying MitM for equality of polynomials, resulting in a linear-time algorithm instead of quadratic-time behaviour displayed by the current client.
- Alignment between CAS servers and the MitM server must be automated. Although manual alignment as described in this case study is usable and enables pinpoint alignments to be made, it is not scalable. Any CAS that aims to be MitM-compatible should develop a representation of its type system in MMT, so that, upon establishing a connection with a new CAS system, the MitM server would be able to automatically align newly accessible functions with symbols from public CDs. This will also enable MMT to act as more than a routing proxy, as it would be able to typecheck the arguments of incoming requests, making the MitM system more rigid.

5 Acknowledgement

I acknowledge financial support from the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541).

References

[1] Isabel M. Araújo et al. *GAP*. URL: `https://gap-system.org`.

[2] Damian Avila et al. *Jupyter*. URL: `https://jupyter.org`.

[3] Wolfram Decker et al. *Singular*. URL: `https://www.singular.uni-kl.de`.

[4] Luca De Feo. *Python SCSCP Module*. URL: `https://github.com/OpenMath/PySCSCP`.

[5] Sebastian Gutsche. *PySingular Module Repository*. URL: `https://github.com/sebasguts/SingularPython`.

[6] Markus Pfeiffer, Dennis Müller, and Victor Vasilyev. *MitM/groups MathHub Repository*. URL: `https://gl.mathhub.info/MitM/groups`.

[7] Florian Rabe and Dennis Müller. *MMT Repository*. URL: `https://github.com/UniFormal/MMT`.

[8] Victor Vasilyev. *MitM Proof of Concept Repository*. URL: `https://github.com/vv20/mitm_proof_of_concept`.