

UNIVERSIDADE ABERTA



Francy

**An Interactive Discrete Mathematics
Framework for GAP**

Manuel Carlos Machado Martins

Doctor's Degree in Computational Algebra

2019

UNIVERSIDADE ABERTA



Francy

**An Interactive Discrete Mathematics
Framework for GAP**

Manuel Carlos Machado Martins

Doctor's Degree in Computational Algebra

Doctoral dissertation supervised by:

Dr João Araújo, Universidade Aberta
Dr James D. Mitchell, University of St Andrews
Dr Markus Pfeiffer, University of St Andrews

2019

Resumo

Em Julho de 2014, uma sessão intitulada *A Proposal of scalable web framework for running GAP on the cloud* foi apresentada no Workshop International em Algebra Computacional em Lisboa. Até ali, a utilização do GAP na Web era uma tarefa difícil, e esta apresentação trouxe algumas ideias novas na tentativa de resolver este problema. Foi então que surgiu um pequeno projeto chamado WebGAP. Em 2015, surge o projeto OpenDreamKit. O projecto OpenDreamKit junta universidades de toda a Europa e visa fortalecer a investigação através da disponibilização de uma plataforma colaborativa virtual. Um dos seus objetivos é enriquecer o ecossistema do Jupyter, fornecendo meios de reprodutibilidade na ciência computacional. Em 2017, a package de software para o GAP Jupyter Kernel é finalmente disponibilizada, tornando possível utilizar o GAP num Web Browser, de forma simples e segura. Foi então identificada a falta de uma package de software para o GAP que permitisse criar interfaces gráficas interactivas, que deu origem a uma package de software para o GAP chamada Francy. Francy é uma package de software para o GAP que permite criar interfaces gráficas interactivas facilitando a representação e exploração de estruturas de dados. Francy é uma package independente de qualquer linguagem de programação ou sistema operativo e permite que virtualmente qualquer outra package de software para o GAP seja transformada numa ferramenta interativa, dando representação gráfica a estruturas matemáticas na forma de grafos ou gráficos topológicos.

Keywords: Visualização, Interação, Gráficos, Matemática, Jupyter, GAP

Abstract

A Proposal of scalable web framework for running GAP on the cloud was presented at the international Workshop on Computational Algebra in Lisbon in July of 2014. At that time, running GAP on the web was proven to be a difficult task, and this solution brought some fresh new ideas to solve it. A small project aroused, it was called WebGAP. In 2015, the OpenDreamKit project started. The OpenDreamKit project brings together universities across Europe and aims to strengthen virtual research. One of its goals is to enrich the Jupyter ecosystem by providing means to reproducibility in computational science. In 2017, the GAP Jupyter Kernel package is finally made available making it possible to run GAP on a web browser, in a simple and secure way. By that time, it was obvious that a package for providing a graphical user interface for GAP was missing. Francy is a package for GAP, and aims to provide graphical interfaces for representation and exploitation of data structures, cross platform wise. Francy allows virtually any GAP package to be transformed into an interactive tool, by giving graphical representation to mathematical structures in the form of topological graphs or charts, making it independent of any particular programming language or operating system.

Keywords: Visualisation, Interaction, Graphics, Mathematics, Jupyter, GAP

Dedication

I am very grateful to all my family, specially to my parents Maria Alice Silva Machado Martins and Jorge Ferreira Martins, and to my friends, specially to Francesca Fusco, for all the support since the first day of this journey,
love you all.

I lost track of time, I had highs, lows and in-betweens... so much has changed
and when I least expected, along came Polly...

Acknowledgements

I am very grateful to Markus Pfeiffer, Pedro A. García-Sánchez, João Araújo, James D. Mitchell, Alexander Konovalov, Sebastian Gutsche and all the GAP community for all the ideas and support during all my dissertation period.

I am also grateful to CoDiMa (CCP in the area of Computational Discrete Mathematics - EPSRC EP/M022641/1, 01/03/2015-29/02/2020) for supporting the attendance at the event Computational Mathematics with Jupyter 2017 in Edinburgh, in which some of this research was done.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Background | 3 |
| 1.2 | Contributions | 4 |
| 1.3 | Thesis outline | 5 |
| 2 | Running GAP on the Web - WebGAP | 7 |
| 2.1 | A scalable web framework for running GAP | 7 |
| 2.2 | Proposed Solution | 8 |
| 3 | IDE tools for GAP | 15 |
| 3.1 | GAP Code Highlighter | 16 |
| 3.2 | GAP Code Linter - Error Checker | 17 |
| 4 | A Graphics API for GAP - Francy | 21 |
| 4.1 | Francy - An Interactive Discrete Mathematics Framework for GAP . . | 22 |
| 4.2 | XGAP vs Francy | 27 |
| 4.3 | Francy Work Environment | 27 |
| 4.4 | Francy Core API | 28 |
| 5 | Packages using Francy | 31 |
| 5.1 | Francy Monoids | 31 |
| 5.2 | Subgroup Lattice | 38 |
| 6 | Conclusion | 43 |
| 6.1 | Contributions | 43 |
| 6.2 | Limitations | 44 |
| 6.3 | Future Work | 45 |
| | Bibliography | 47 |
| | Appendices | 53 |
| A | Francy User Manual | 55 |

List of Figures

| | | |
|------|---|----|
| 2.1 | WebGAP conceptual high level diagram | 10 |
| 2.2 | WebGAP Portal | 11 |
| 2.3 | WebGAP Marketplace | 11 |
| 3.1 | Highlighting features on Jupyter. | 17 |
| 3.2 | Linting and highlighting features on Jupyter. | 19 |
| 4.1 | Directed graph of all subgroups of S_3 | 24 |
| 4.2 | A simple example of an interactive graph | 30 |
| 5.1 | Factorizations graph using Graphviz-Renderer with circ engine | 33 |
| 5.2 | Factorizations Eliahou Graph using D3 Renderer | 34 |
| 5.3 | Rosales Graph using D3 Renderer | 35 |
| 5.4 | Hasse diagram of oversemigroups using Graphviz Renderer | 36 |
| 5.5 | Tree of sons of numerical semigroups using D3 Renderer | 36 |
| 5.6 | Hasse Diagram using Graphviz Renderer | 37 |
| 5.7 | Tree of Gluing using D3 Renderer | 38 |
| 5.8 | Subgroup lattice of S_4 , using D3 Renderer | 39 |
| 5.9 | All subgroups lattice of S_4 , using D3 Renderer | 40 |
| 5.10 | Subgroup lattice of S_4 , subgroup properties | 41 |

Code Listings

| | | |
|------|--|----|
| 2.1 | WebGAP API base request | 9 |
| 2.2 | WebGAP API base response | 9 |
| 3.1 | GAP code regular expressions | 16 |
| 3.2 | Part of the GAP EBNF grammar for ANTLR4 | 18 |
| 4.1 | GAP code to display directed graph of all subgroups of S_3 | 25 |
| 4.2 | A simple example of an interactive graph | 29 |
| 5.1 | Create a numerical semigroup and factorizations | 33 |
| 5.2 | Factorizations of numerical semigroup result | 33 |
| 5.3 | Create a numerical semigroup and draw the over semigroups | 35 |
| A.1 | Jupyter extension installation | 56 |
| A.2 | Jupyter extension enable | 56 |
| A.3 | Francy simple Callback with no arguments | 58 |
| A.4 | Francy callback with one required argument | 58 |
| A.5 | Francy callback with one known argument | 58 |
| A.6 | Francy callback with default trigger type | 58 |
| A.7 | Francy callback with custom trigger type | 58 |
| A.8 | Francy no operation callback | 59 |
| A.9 | Francy callback trigger function | 59 |
| A.10 | Francy simple canvas | 62 |
| A.11 | Francy simple line chart | 67 |
| A.12 | Francy simple scatter chart | 67 |
| A.13 | Francy simple bar chart | 67 |
| A.14 | Francy simple directed graph | 73 |
| A.15 | Francy simple undirected graph | 74 |

Acronyms

API Application Programming Interface.

CAS computer algebra system.

EBNF Extended Backus-Naur Form notation.

GAP Groups, Algorithms, Programming.

GUI Graphical User Interface.

IaaS Infrastructure as a Service.

IDE Integrated Development Environment.

JSON JavaScript Object Notation.

MIME A Multipurpose Internet Mail Extensions.

PaaS Platform as a Service.

REPL Read–Eval–Print Loop.

SaaS Software as a Service.

Chapter 1

Introduction

We live in an era where we expect to find everything online. We can see the proliferation of internet services around all sorts of areas, including mathematics. By the time I have started studying Group and Semigroup Theory I was introduced to Groups, Algorithms, Programming (GAP) - a System for Computational Discrete Algebra. GAP is a powerful tool and contains thousands of algorithms to aid the resolution of common problems in Group and Semigroup theory. Since I started using it, some limitations were just obvious. I found it very peculiar that the installation of this software required some knowledge of software building tools, making it not trivial for most users. All the procedure involved, with core compilation and packages compilation, was just too much for the regular personal computer user. Not to mention that updates would require similar amount of work, making users stick with old working versions just to avoid the hassle of maintenance.

In the pursuit of a solution for this problem, I have created a small proof of concept using web-ssh, providing online access to a GAP shell. In July of 2014, at the Workshop on Computational Algebra in Lisbon, a small talk titled *"Proposal of scalable web framework for running GAP on the cloud"* brought into light a project called WebGAP[1, 2]. The main goal of that project was to provide a centralised environment where users could easily use GAP [3], by writing, executing and sharing code, without any need for installation, configuration or upgrades. This was an ambitious project and indeed called the attention of many users, including the company AHP [4] which kindly offered to host the web application.

WebGAP is a simple web application using a social log-in such as Google or Facebook and allows users to access a GAP shell, running on a remote server, through a web browser. This simple concept has also the potential to be used to provide, by the same means, access to many other applications, making it easier to learn, use and adopt new software. WebGAP was under active development until late 2015, but without any major release.

By the end of that same year, in September 2015, a new project called OpenDreamKit [5] was announced. This project, as described in the original abstract [6], aims to:

CHAPTER 1. INTRODUCTION

"OpenDreamKit will deliver a flexible toolkit enabling research groups to set up Virtual Research Environments, customised to meet the varied needs of research projects in pure mathematics and applications, and supporting the full research life-cycle from exploration, through proof and publication, to archival and sharing of data and code. OpenDreamKit will be built out of a sustainable ecosystem of community-developed open software, databases, and services, including popular tools such as LinBox, MPIR, Sage(sagemath.org), GAP, PARI/GP, LMFDB, and Singular. We will extend the Jupyter Notebook environment to provide a flexible UI. By improving and unifying existing building blocks, OpenDreamKit will maximise both sustainability and impact, with beneficiaries extending to scientific computing, physics, chemistry, biology and more, and including researchers, teachers, and industrial practitioners."

This was clearly a step forward, and would tackle some of the aspects WebGAP was trying to solve. I was introduced to this project by Markus Pfeiffer, a research fellow in the School of Computer Science at University of St Andrews. Pfeiffer invited me to work on the GAP interface with Jupyter, to provide a rich online environment where users could play with GAP. Within the OpenDreamKit project, GAP is a core component and St. Andrews University is responsible for it [7].

My involvement in the OpenDreamKit from the beginning allowed me to merge WebGAP on this project. Following to some discussion and with more insight from the GAP community, aligned with the OpenDreamKit objectives, a Graphics API for GAP was identified as a must have component. The Graphics API would enable interactivity by providing a graphical representation of mathematical structures, enabling, among others, learning by visualisation.

A modern Graphics Application Programming Interface (API) for GAP, running on web technologies, is now a reality and is called Francy. Francy is my main contribution and is part of the deliverable D4.7 [8] and D4.16 [9] of OpenDreamKit project.

Most of the work produced is based on the existing package for GAP called XGAP. Francy's API tries to match, when possible, with XGAP so users familiar with it can start using Francy with minimum effort. An attempt to reuse XGAP was tested at some point, but without any success.

From here, Francy gives light to many other packages, including FrancyMonoids and SubgroupLattice. Other projects, inspired by Francy, are now available, such as Jupyter-Viz [10], a package that enables the usage of any JavaScript toolkits for drawing charts and plots directly from GAP. Some material has been produced for teaching purposes, by using the GAP Jupyter Kernel or Francy, e.g. Pedro A. García-Sánchez at the University of Granada in Spain [11]. The author of the project GAP.APP [12], which is XGAP ported to MacOS, is experimenting with the portability to Francy.

Multiple modules developed for WebGAP were reused and were incorporated into Francy or in the new developments for Jupyter. The GAP mode, which provides

code highlighting and indentation in Jupyter is distributed with the GAP Jupyter Kernel package. The GAP Linter that provides GAP support on Jupyter and Brackets IDE, namely code highlighting, code indentation and code linting is based on a simple grammar for GAP built using ANTLR4 [13].

During this period a Microsoft Research was granted [14]. This grant ensured computing resources for the whole project including the test platform CloudGAP¹ [15] for one year long.

1.1 Background

GAP

GAP stands for *Groups, Algorithms, Programming (GAP)* and is the world's most popular computational algebra system for computational group theory.

GAP has a strong community and is actively developed. It was started at Lehrstuhl D für Mathematik, RWTH Aachen in 1986.

GAP is used by students, teachers, researchers and everyone willing to learn Group / Semigroup theory.

XGAP

The graphics package XGAP [16] is a reference in GAP. XGAP is integrated with the Unix X-Window System, which provides a framework for a Graphical User Interface (GUI), and includes a wide range of mathematical functionality focused primarily on the lattice of subgroups of a group. Notably, the project GAP.APP is XGAP ported to MacOS [12].

Jupyter

Jupyter, as stated in the project page [17], *"is an open-source web application that allows you to create and share documents that contain live code, equations, visualisations and narrative text"*. It is possible to extend Jupyter's functionality either by implementing notebook extensions [18] or adding new kernels [19].

A kernel in Jupyter is a software that makes the connection between Jupyter and the underlying language software introspecting user's code on a notebook. GAP Jupyter Kernel [20] is the base package that enables GAP to run on Jupyter.

A notebook extension in Jupyter is a software that provides extra functionality at notebook level, e.g. a new Multipurpose Internet Mail Extensions (MIME) Type [21] support, a new programming language mode [22], a new menu entry, etc. Francy relies on this functionality, to create an interactive representation of data, by creating a new MIME Type *application/vnd.francy+json*. GAP Jupyter Kernel relies

¹CloudGAP was an in-house installation of Jupyter for use at St. Andrews University in Scotland.

on this functionality to provide the GAP programming language mode, enabling code highlight and indentation.

1.2 Contributions

WebGAP

WebGAP stands for "*GAP on the Web*", and is a project aiming to provide a simple and fast way to start playing around with GAP.

This project started as a simple web application allowing access to the GAP language command shell - Read-Eval-Print Loop (REPL), it evolved to a containerised solution using Docker that could provide means to access many other applications distributed in its own *Applications Marketplace*.

Many small modules developed on this project were later reused to provide a richer learning environment on Jupyter.

GAP IDE Tools

The GAP IDE Tools, namely the highlighting and linting modules, provide a richer environment on Jupyter to write GAP code.

Francy

Starting from simple concepts of drawing and graph theory, Francy is inspired by XGAP.

Based on XGAP, Francy [23] rises as a graphics package for GAP for the web. Francy brings some new concepts and separates completely the graphics creation and the GAP code by creating a semantics data layer. This approach decouples GAP from any GUI framework, allowing one to display graphics using any programming language, on any operating system platform.

Francy uses renderers to display graphics. These renderers are plugins that implement a visualisation framework that gives a representation to the semantic model. At the moment Francy is distributed with D3.js [24], Graphviz [25] and Vis.js [26] renderers. This feature allows users to switch between renderers, at any time, providing different representations of the same mathematical structure for exploitation.

1.3 Thesis outline

Notwithstanding this Introduction chapter and the Conclusion chapter, this thesis is divided into four chapters.

Chapter 2, Running GAP on the Web - WebGAP, presents all the efforts to make GAP running on the web. It presents a web application allowing users to interact with a GAP console, how it could be extended to provide access to other similar applications, how it scales and how security can be achieved.

Chapter 3, IDE tools for GAP, introduces some tools developed for WebGAP and later integrated into Jupyter to help users writing and interacting GAP code.

Chapter 4, A Graphics API for GAP - Francy, introduces a Graphical User Interface (GUI) framework created to enrich GAP, providing an interactive environment for visualisation and exploitation of mathematical structures taking advantage of open source JavaScript rendering frameworks for the web. This chapter shows the core concepts of `francy` and ideas on how it can be used to enhance the usability of existing GAP packages.

Chapter 5, Packages using Francy, shows some GAP packages developed using `francy`, namely, Francy Monoids and Subgroup Lattice.

Chapter 2

Running GAP on the Web - WebGAP

This chapter contains unpublished results presented in 2014 at the International Workshop on Computational Algebra, Lisbon - Portugal. An updated version of this project was later, in 2015, presented at the Software Engineering Seminar, Barcelos - Portugal.

This chapter presents a scalable web-based service, available through WebGAP - <https://portal.webgap.eu> - which utilises node.js for client-server integration. The service allows access to GAP - <http://gap-system.org> - through a web-browser using web-based SSH. It provides a collaborative environment and aims to ease and accelerate GAP usage.

It is presented the design and architecture of the web framework along with an overview on security and privacy. This framework can also be extended to provide access to other UNIX-based tools without the need to code-change, through the use of an applications market. Examples of possible extensions are also mentioned in this chapter.

Some WebGAP components have been reused and integrated in new developments for the OpenDreamKit project. The project is not active at the moment.

This work has been supported by João Araújo - Universidade Aberta and James Mitchell - University of St Andrews.

2.1 A scalable web framework for running GAP

As described in the official GAP website[3]: *GAP is a system for computational discrete algebra, with particular emphasis on Computational Group Theory. GAP provides a programming language, a library of thousands of functions implementing algebraic algorithms written in the GAP language as well as large data libraries of algebraic objects.*

The purpose of this project comes from the need to access GAP through a web-based service giving the possibility to use it from anywhere using any kind of

CHAPTER 2. RUNNING GAP ON THE WEB - WEBGAP

mobile device with internet access. This will abstract users from technical aspects such as installation and/or upgrade procedures.

The internet has changed the way people work. It gives users the possibility to access all kind of information whether at work, travelling or at home. Internet availability has increased in order to fulfil user demands, and mobile devices, such as smart-phones and tablets, are mostly responsible for this. The increase of such devices usage, raises some questions: *How can we make a software portable with minimum cost and effort? How can we provide it?*

Installation of desktop applications requires some computer skills and, depending on the user, a fair amount of time. Upon installation, the user needs to maintain the application with new upgrades as they become available. The resources needed to run such application can sometimes be a problem and have some impact on the normal usage of a personal computer. In contrast, web based services deployment is done centrally on the server, without any intervention required by the user, becoming immediately available and accessible, e.g., through a web-browser.

The rise of Cloud Computing, naming Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), turned business innovation accessible to most companies, leading them to re-think business models, and thus transforming desktop applications into web-based services, with the result of being available on-demand with reasonable usage-based cost [27, 28].

With this project, we answer the two questions. First, by providing a service we remove the extra effort and cost on producing different flavors of the software. Second, by using a web-based service, we ensure that only a web browser and an internet connection are required to use the software.

The next section exposes the design and architecture, along with a technical description and the features offered for a solution to provide access to GAP software through a web browser.

2.2 Proposed Solution

A system ability to grow in resources based on the amount of work being handled at a certain point is called scalability. Scalability is a complex concept and can be evaluated in multiple and different ways. As *"being scalable"* is one of the main requirements, this project started to be developed having in mind the ability to grow and shrink resources in order to adjust to its real usage. Cloud computing offers, out-of-the-box, the possibility to scale, horizontally or vertically [29], depending on the service model chosen, IaaS, PaaS or SaaS.

The solution presented follows a broker architecture pattern. This architecture favours scalability [30], therefore it allows the system to adapt on resources

whenever needed, by scaling horizontally. The implementation follows a modular-pattern, *"write simple parts connected by clean interfaces"*, in order to isolate implementation and ease maintenance [31]. All the system is built on top of a Node.js stack. Node.js is a runtime built on Google Chrome V8 Javascript Engine and is mainly used in real time applications and is a non-blocking event driven framework that allows one to build fast and scalable web services [32].

In terms of message exchanging between client and server, all the messages exchanged are JSON based, defined by a JSON-Schema, over HTTPS. A request message example is presented in the code listing 2.1, similarly a response message is presented in the code listing 2.2.

```
{
  "token": {
    "host": "",
    "creator": "",
    "issuedAt": "",
    "expiresAt": ""
  },
  "request": {
    "name": "",
    "data": {}
  }
}
```

Listing 2.1: WebGAP API base request

```
{
  "token": {
    "host": "",
    "creator": "",
    "issuedAt": "",
    "expiresAt": ""
  },
  "response": {
    "name": "",
    "data": {}
  }
}
```

Listing 2.2: WebGAP API base response

All token and data fields are encoded as Json Web Tokens (JWT) using RS256 algorithm [33]. JWT is a standard method for representing claims to be transferred between two parties. The RS256 stands for RSA Signature with SHA-256, and is an asymmetric algorithm which uses a private/public key pair. JSON stands for JavaScript Object Notation and has the same interoperability potential as XML, but its structure is simpler and lightweight and is commonly used on web-services [34].

CHAPTER 2. RUNNING GAP ON THE WEB - WEBGAP

Access to GAP console is given by the means of a web-based SSH terminal. SSH is one of the most successfully secure technologies used nowadays [35], it stands for Secure Shell and is a cryptographic network protocol for secure data communication mainly used for access to shell accounts in UNIX systems. Web-based SSH is implemented mainly in JavaScript and HTML and uses web-sockets to communicate with SSH servers. This technology allows to serve virtually anything that runs in a UNIX shell and thus it is used to serve a GAP console. Web-sockets is a technology that makes possible a two-way interactive session between a client, or web browser, and a server.

The diagram in figure 2.1 represents the framework conceptual high level diagram of the proposed solution to run GAP on the cloud.

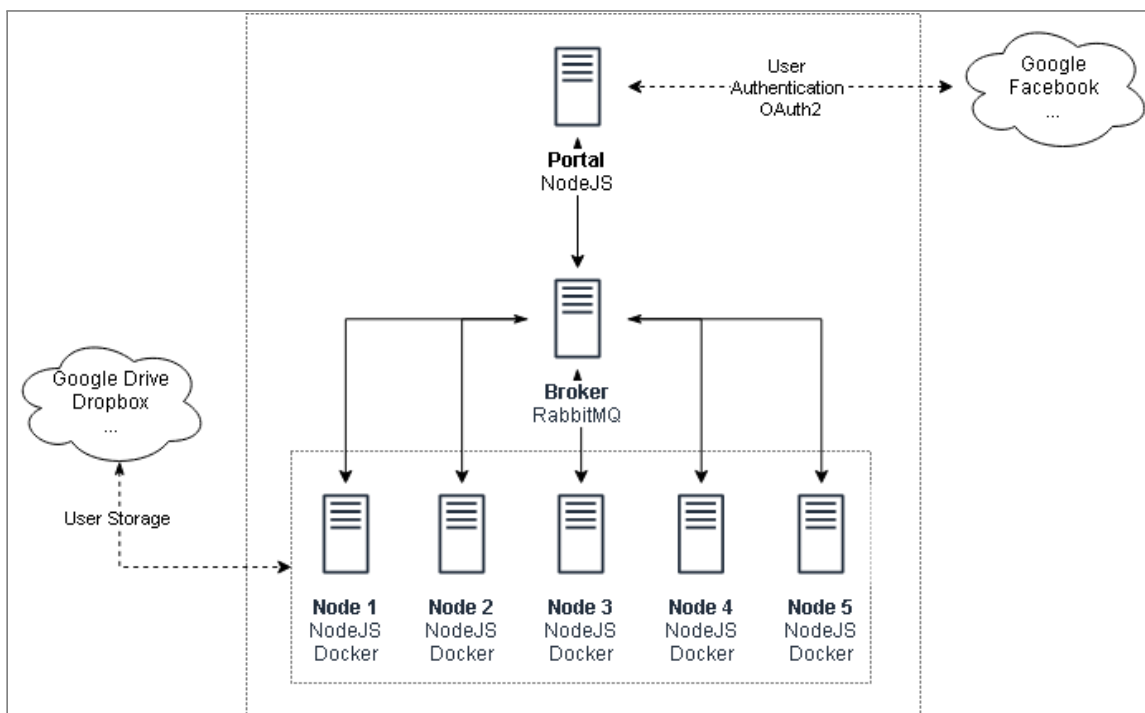


Figure 2.1: WebGAP conceptual high level diagram

The framework integrates 3 layers with basic systems: one **Portal**, one **Broker** and a set of worker **Nodes**, represented on the diagram in figure 2.1.

a) Portal

The Portal is responsible for serving all the services through a web GUI. The web GUI, built on top of Node.js, is simple and intuitive and allows the user to rapidly adapt to the work environment. The authentication mechanism provided by the portal relies on OAuth2 protocol. In order to use the system, the user must authenticate using one of the supported providers, either Google, Facebook or Github. The avoidance of security issues is the principal reason for using such authentication mechanism, so delegating all the responsibility to a bigger and reliable provider ensures no critical data handle on the application side [36]. The system can be extended to provide authentication based in Single Sign On systems used,

CHAPTER 2. RUNNING GAP ON THE WEB - WEBGAP

for instance by universities. In terms of authorisation, all users have the same role and there is a profile associated with each user which contains all the information needed to instantiate the services. In the future, storage will also be available by third party providers, and once again, delegating security and reliability to a thrust provider of users' choice. This will ensure also out-of-the-box features like sharing and concurrency, since it will be handled by the provider.

Figures 2.2 and 2.3 show, respectively, the portal interface to the GAP console and the applications marketplace.

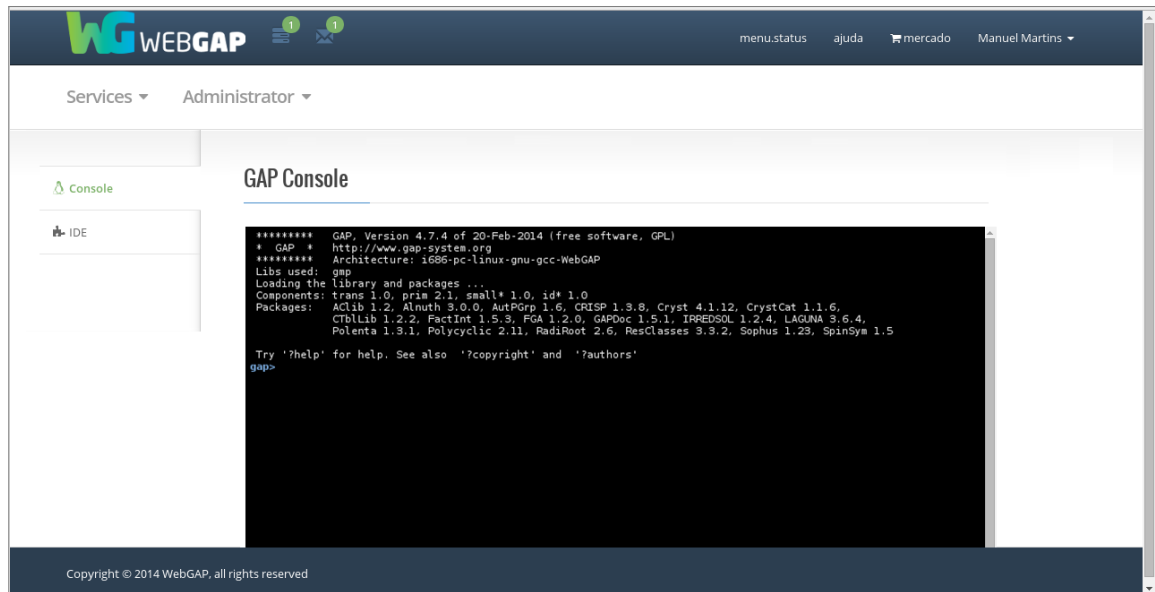


Figure 2.2: WebGAP Portal

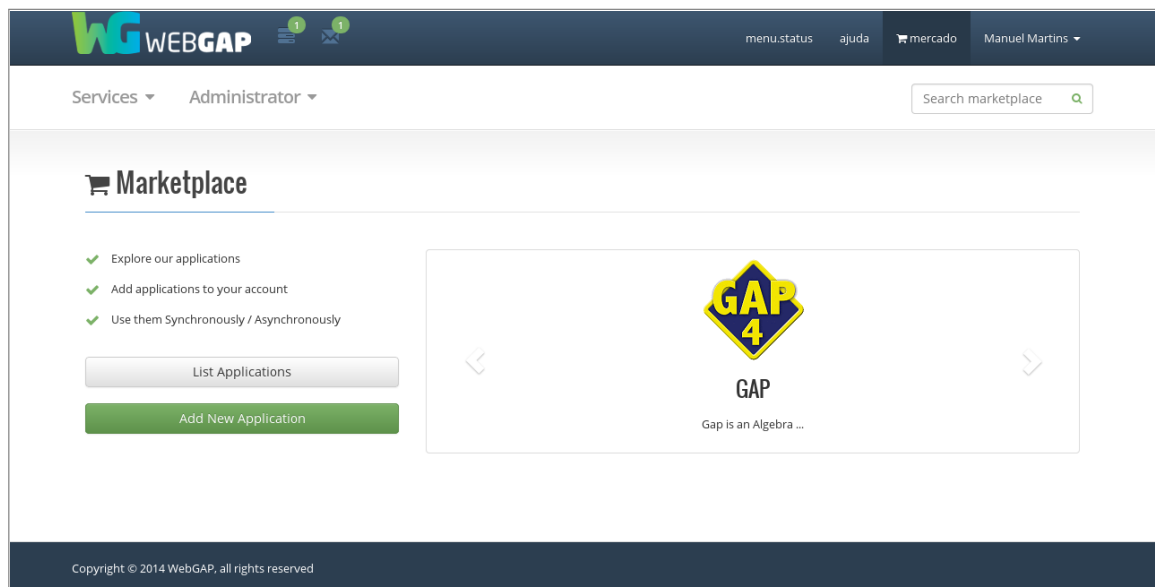


Figure 2.3: WebGAP Marketplace

b) Broker

The broker handles all the communication between the portal and the nodes. The creation of this intermediate system relies on the simple principle of modularity and abstraction of concepts. Filtering and cache support is provided in order to ensure messaging reliability.

RabbitMQ [37] is a queuing system and allows applications to subscribe queues and transfer messages onto. RabbitMQ helps decoupling applications and will also ensure asynchronous processing, or work queues for long running processes such as proofs on Prover9.

c) Node

The Node is responsible for serving GAP instances. It is responsible for handling security measures, such as client authentication and authorisation. Depending on the necessary resources to instantiate a new service, a Node will reply with the HTTP uri address, where the service is exposed, along with an access token. This response is used by the Portal to provide access to the web-based SSH console.

The web-based SSH console is based in Wetty (an hterm emulation tool written in JavaScript). *"HTML Terminal"*, or hterm, is an xterm-compatible terminal emulator written entirely in JavaScript and uses Chrome OS terminal emulator. It is intended to be fast enough and correct enough to compete with native terminals such as xterm, gnome-terminal, konsole and Terminal.app. hterm is only a terminal emulator. In order to make this terminal communicate with GAP there is a small Node.js implementation based in web-sockets [38]. In order to secure and ensure the integrity of the system, all the services run with low privileges, which means the service has low access to system resources. There are filters implemented on the web-sockets solution that prevents the execution of some low level GAP commands. Currently these services are installed in Docker [39] images so that they run in isolation from other processes.

Docker performs operating-system-level virtualisation and uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, making it easier to run applications in an isolated environment. This will create a "sandboxing" environment where applications run isolated from other applications, making it easier to instantiate multiple instances of a specific application.

A Node can be extended and can provide access to other services, by implementing new Docker images. It can be used, for instance, to provide access to a Prover9 instance. Access to new tools can be provided by means of a console or simply an asynchronous remote execution.

Conclusion

In this chapter a solution to run GAP on the cloud was presented. It is based on the broker architecture pattern and it uses web-based SSH technology to provide

CHAPTER 2. RUNNING GAP ON THE WEB - WEBGAP

access to GAP console. This project has the potential to attract researchers and students in the Algebra fields whom approach to the Group / Semigroup theory. This is both because it provides them with a new way to access and work with GAP, and it is mainly focus in usage aid and collaboration.

Although some new features will appear in the future, like major browser support - since only Chrome browser is supported -, the Integrated Development Environment for GAP language to increase the code productivity by introducing error highlighting and snippets, the integration with cloud storage to bring flexibility and collaboration between users, the main goal will remain intact: provide access to GAP through a web-based SSH Console.

Future Work

Many improvements can be done in terms of scalability, security and features.

WebGAP could be used as a tool with means to execute long run tasks, asynchronously, making use of computational resources for, for instance, running some long process and demanding proof software tool like Prover9/Mace4.

Chapter 3

IDE tools for GAP

This chapter contains published results part of the GAP Jupyter Kernel distribution and the OpenDreamKit Deliverables D4.4.

The focus of the OpenDreamKit, as mentioned before, is to enrich the Jupyter ecosystem with the most popular mathematical tools, where GAP is provided by the GAP Jupyter Kernel, hence the inclusion of the IDE tools along with it.

Most programming languages have an Integrated Development Environment (IDE) support for features such as code highlighting, code linting, auto-completion, code snippets, debugging, etc. These features provide an intuitive and productive environment that helps developers write code faster, more efficiently and less error prone. Of course, popular programming languages with huge communities tend to evolve quicker and thus have wider IDEs support.

Following the cloud trend, many IDEs are available online and can be used freely. Brackets [40], Cloud9 [41] and Atom [42] are examples of IDEs that can be used either as a service on the cloud or as a desktop application. New technologies, mostly driven by modern mobile requirements, are used to *transform* web applications into desktop applications [43]. This enables a simple way of keeping a single code base for multiple distributions.

In order to ease the usage of GAP language, and similarly to what is offered by most IDEs for main programming languages, some of these features have been developed to enrich its usage within the Jupyter environment.

GAP is a programming language and thus it follows a grammar. A grammar is a set of rules forming a language structure. Following such a grammar facilitates the creation of tools such as code highlighting, code linting, auto-completion and code-snippets.

From this principle two projects rise, originally developed for WebGAP and later migrated to Jupyter extensions, aiming to help users writing GAP code.

3.1 GAP Code Highlighter

This project aims to provide a syntax highlighter for GAP code. It was first developed and integrated with Cloud9 [41] IDE, during WebGAP development, and was later migrated to CodeMirror [22] which is the default code manager on Jupyter.

Syntax highlighting is a feature that helps better visualising, reading and understanding a language source code by using colours, different fonts and indentation, to display the text according to the properties of that same code language.

Syntax highlighting is achieved using regular expressions. Regular expressions are sequences of characters to define a search pattern. Within this project, each regular expression is used to identify the different properties of the language such as *keywords*, *comments*, *built-in functions*, *operators*, *numbers* and *variables*. Code listing 3.1 shows the regular expressions used to identify GAP code. The full JavaScript module can be seen on the project Jupyter Kernel [20] within the file *JupyterKernel/etc/gap-mode/gap.js*.

```
var keywords = /\bAssert\b|\bInfo\b|\bIsBound\b|\bQUIT\b|\bTryNextM
→ ethod\b|\bUnbind\b|\band\b|\batomic\b|\bbreak\b|\bcontinue\b|\b
→ do\b|\belif\b|\belse\b|\bend\b|\bfalse\b|\bfi\b|\bfor\b|\bfunct
→ ion\b|\bif\b|\bin\b|\blocal\b|\bmod\b|\bnot\b|\bod\b|\bor\b|\bq
→ uit\b|\breadonly\b|\breadwrite\b|\brec\b|\brepeat\b|\breturn\b|
→ \bthen\b|\btrue\b|\buntil\b|\bwhile\b/;
var all = /(?:.)/;
var comment = /(?:#.*/);
var blockLiterals = /(?:"")/;
var literals = /(?:")|(?:')/;
var numbers = /(?:\d*\.\d+(?![\w@\\]))/;
var variables = /(?:\\[(),.]?|[\w@+])/;
var properties = /(?:\+|-|\\*|\\/|\\^|~|!\\.|=|<>|<|<=|>|>=|!\\[|:=|\\.|\\
→ \\.|->|,|;|!{\\[|]|{\\}|\\(\\)|:)/;
// indent next line
var indentTokens = /(?:\bfunction\b|\bif\b|\brepeat\b|\bwhile\b|\ba
→ tomic\b|\bfor\b)/;
// dedent current line
var dedentTokens = /(?:\bod\b|\bend\b|\bfi\b|\buntil\b)/;
// dedent current line, but keep indent next line
var partiallyDedentTokens = /(?:\belse?\b|\belif\b)/;
```

Listing 3.1: GAP code regular expressions

Within Jupyter, identifying GAP code is done by the GAP Jupyter Kernel, that when active, marks input cells on that Jupyter notebook with the MIME type *text/x-gap*. This MIME type is aligned within the kernel and the code highlighter so that the content is rendered accordingly. A MIME type is a standard that indicates the nature and format of a document, file, or assortment of bytes. This standard is very useful as it is commonly used as HTTP headers, *Accept* and *Content-Type*

more precisely, to specify what are the MIME types expected by the client and the MIME type of the response being sent from the server, within a transaction.

This project is distributed with the Jupyter Kernel package for GAP and can be used on Jupyter [20]. It can be easily ported to any other system, that relies on JavaScript, to enable code highlighting. Figure 3.1 shows this functionality on Jupyter.

The code highlighter improves code readability and helps giving context to GAP code.

```
DrawEliahouGraph:=function(n,s)
  local graph, canvas, f, fs, c, nf, i, p;

  f:=FactorizationsElementWRTNumericalSemigroup(n,s);
  graph:=Graph(GraphType.UNDIRECTED);
  nf:=Length(f);
  fs:=[];
  for i in [1..nf] do
    fs[i]:=Shape(ShapeType!.CIRCLE, Concatenation("(" ,JoinStringsWithSeparator(f[i],","),")"));
    SetLayer(fs[i],Sum(f[i]));
    SetSize(fs[i],1);
    Add(graph,fs[i]);
  od;
  c:=Cartesian([1..nf],[1..nf]);
  c:=Filtered(c,p->p[1]<p[2] and f[p[1]]*f[p[2]]<>0);
  for p in c do
    Add(graph,Link(fs[p[1]],fs[p[2]]));
  od;
  canvas:=Canvas("Eliahou graph");
  Add(canvas,graph);
  return Draw(canvas);
end;
```

Figure 3.1: Highlighting features on Jupyter.

3.2 GAP Code Linter - Error Checker

This project aims to provide a code linter for GAP. The linter analyses GAP code and flags programming errors, bugs and stylistic errors [44] before its execution. This functionality helps the developer saving a huge amount of time as the linting occurs before code execution. The linting is done using a parser generated by ANTLR4 [13] and a GAP grammar, created for this project.

ANTLR4, as stated in the official documentation, *"is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It is widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build parse trees and also generates a listener interface (or visitor) that makes it easy to respond to the recognition of phrases of interest"* [45].

An unofficial and incomplete grammar for GAP has been created by myself and is defined in EBNF [46]. The GAP grammar has been created as a proof-of-concept, and not being the main focus of this dissertation, does not cover all the language structure. Rather, it provides a starting point, or a good indication, on how to build a tool to parse GAP code. A part of the GAP grammar can be seen in code listing 3.2, retrieved from the project GAP-lint [44] within the file *gap-lint/master/src/gap.g4*.

CHAPTER 3. IDE TOOLS FOR GAP

```
grammar gap;
parse
  : block EOF
  ;
block
  : statement+
  ;
statement
  : expression SemiColon SemiColon?
  | ifStatement SemiColon SemiColon?
  | forStatement SemiColon SemiColon?
  | whileStatement SemiColon SemiColon?
  | doStatement SemiColon SemiColon?
  | repeatStatement SemiColon SemiColon?
  | TrippleQuote
  ;
functionDecl
  : Function OParen idList? CParen (statement|Local idList
  → SemiColon)+ End
  ;
shortFunctionDecl
  : OBrace idList? CBrace ShortHandFunction expression OParen
  → idList? CParen
  ;
doStatement
  : Do block? Od
  ;
ifStatement
  : ifStat elseIfStat* elseStat? Fi
  ;
ifStat
  : If expression Then block?
  ;
elseIfStat
  : Elif expression Then block?
  ;
elseStat
  : Else block?
  ;
/** ... */
```

Listing 3.2: Part of the GAP EBNF grammar for ANTLR4

The GAP linter is based on the grammar shown in the code listing 3.2 and can be used within Jupyter or the Brackets IDE [40] where it is available as an extension through the Brackets Extension Manager. The portability of this module to any other IDE or any other web framework is trivial and should be achieved with minimal effort.

The figure 3.2 shows a typical example of a syntax error, a missing parenthesis, being flagged by the linter.

```

In [19]: 1 canvas := Canvas("Callbacks in action");;
          2 SetTexTypesetting(canvas, true);;
          3 SetHeight(canvas, 450);;
          4
          5 graph := Graph(GraphType.UNDIRECTED);;
          6 graph := Graph(GraphType.UNDIRECTED);;
          7 graph := Graph(GraphType.UNDIRECTED);;
          8 Add(canvas, graph);;
          9
          10 HelloWorld := function(name)
          11     Add(canvas, FrancyMessage(Concatenation("Hello, ", name)));
          12     return Draw(canvas);
          13 end;;
          14
          15 callback1 := Callback(HelloWorld);;
          16 arg1 := RequiredArg(ArgType.STRING, "Your Name?");;
          17 Add(callback1, arg1);;
          18
          19 menu := Menu("Example Menu Holder");;
          20 menu1 := Menu("Hello Menu Action", callback1);;
          21 Add(menu, menu1);;
          22
          23 Add(canvas, menu);;
          24 Add(canvas, menu1);;
          25 Add(shape, menu1);;
          26
          27 Draw(canvas);
  
```

The tooltip shows two error messages:

- missing ';' at '('
- extraneous input ';' expecting '{', ',', '}'

Figure 3.2: Linting and highlighting features on Jupyter.

The code linting improves developer's efficiency as well as overall code quality, by providing a static validation of the written code and detecting programming errors, bugs and stylistic errors, before its execution.

Chapter 4

A Graphics API for GAP - Francy

This chapter contains published results [47] presented in 2018 at the International Congress of Mathematical Software, South Bend - USA.

Francy was officially released on 1st of November 2018 and is now distributed by default on every release of GAP, since version 4.10.0.

Francy is the result of the work produced while building a GUI framework for GAP and it is best described in the publication below.

The authors of the original paper, transcribed in this chapter, are very grateful to James D. Mitchell, João Araújo, Pedro A. García-Sánchez and Francesca Fusco for their suggestions that led to a much improved version of the paper in this section.

The first author is grateful to CoDiMa (CCP in the area of Computational Discrete Mathematics - EPSRC EP/M022641/1, 01/03/2015-29/02/2020) for supporting the attendance at the event Computational Mathematics with Jupyter 2017 in Edinburgh, in which some of this research was done. The second author has received funding from the European Union project Open Digital Research Environment Toolkit for the Advancement of Mathematics (EC Horizon 2020 project 676541, 01/09/2015-31/08/2019).

4.1 Francy - An Interactive Discrete Mathematics Framework for GAP

Data visualisation and interaction with large data sets is known to be essential and critical in many businesses today, and the same applies to research and teaching, in this case, when exploring large and complex mathematical objects.

GAP is a computer algebra system for computational discrete algebra with an emphasis on computational group theory. The existing XGAP package for GAP works exclusively on the X Window System. It lacks abstraction between its mathematical and graphical cores, making it difficult to extend, maintain, or port. In this paper, we present Francy, a graphical semantics package for GAP. Francy is responsible for creating a representational structure that can be rendered using many GUI frameworks independent from any particular programming language or operating system.

Building on this, we use state of the art web technologies that take advantage of an improved REPL environment, which is currently under development for GAP. The integration of this project with Jupyter provides a rich graphical environment full of features enhancing the usability and accessibility of GAP.

Introduction

By providing a mechanism for quickly demonstrating a topic, or result, visual learning has been proven effective and advantageous. It helps with engagement and allows students to look at problems in a different way [48].

In mathematics, especially in group theory, having a graphical representation of certain structures is invaluable when formulating conjectures and counterexamples, and when analysing data. GAP is a computer algebra system (CAS) focused on computational group theory and it helps to explore algebraic structures and solve a variety of problems [3]. The primary existing package for GAP, which provides facilities displaying graphics and visualisation of mathematical data structures, is XGAP. This package is integrated with the Unix X-Window System, which provides a basic framework for a Graphical User Interface (GUI), and includes a wide range of mathematical functionality focused on the lattice of subgroup of a group [16]. Further such packages include Interactive Todd Coxeter (ITC) which was developed using XGAP and provides an interactive environment for exploring coset enumerations [49]. GAP.APP is another project based on XGAP and it provides a native Macintosh interface for GAP [12]. All of these projects enable GAP to be used as a tool to visualise objects with computer graphics.

Technology evolves quickly and today multiple web platforms allow users to experience, learn, and share in a simple and fast paced environment. Jupyter is one of these projects and, as mentioned on the official website [17], aims “*to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages*”, leveraging learning processes

and the way people share their work. The purpose of the OpenDreamKit project is to provide a framework for the advancement of mathematics in Europe, as part of the Horizon2020 European Research Infrastructure, and Jupyter is a core component of OpenDreamKit [5]. Jupyter allows a centralised system for the dissemination of content and uses an intuitive interface where users interact with notebooks containing live code, equations, visualisations and narrative text.

Francy [23] arose from the necessity of having a lightweight framework for building interactive graphics, generated from GAP, running primarily on the web, primarily in a Jupyter Notebook. An initial attempt to re-use XGAP and port it was made, but the lack of a standardised data exchange format between GAP and the graphics renderer, and the simplistic initial requirements of the project were the basis for the creation of a new GAP package.

Functionality

Francy provides an interface to draw graphics using objects. This interface is based on simple concepts of drawing and graph theory, allowing the creation of directed and undirected graphs, trees, line charts, bar charts and scatter charts. These graphical objects are drawn inside a canvas that includes a space for menus and to display informative messages. Within the canvas it is possible to interact with the graphical objects by clicking, selecting, dragging and zooming.

In terms of interaction with the kernel, we use callbacks which allow the execution of functions in GAP from the graphical objects. A callback holds the function signature and any arguments that it requires. If a callback requires user input, a modal window will be shown before the execution of the function.

Applications

Francy does not provide any mathematical functionality as it is intended to be used by other mathematical software packages. Existing GAP packages can be easily ported to use it. Francy has potentially many applications and can be used to provide a graphical representation of data structures, allowing one to navigate through and explore properties or relations of these structures. In this way, Francy can be used to enrich a learning environment where GAP provides a library of thousands of functions implementing algebraic algorithms as well as large data libraries of algebraic objects.

In the code listing 4.1 we show a simple usage of Francy to display interactively the directed graph of all subgroups of the Symmetric Group S_3 , using the GAP package Digraphs [50]. The code snippet result can be seen in figure 4.1.

CHAPTER 4. A GRAPHICS API FOR GAP - FRANCY

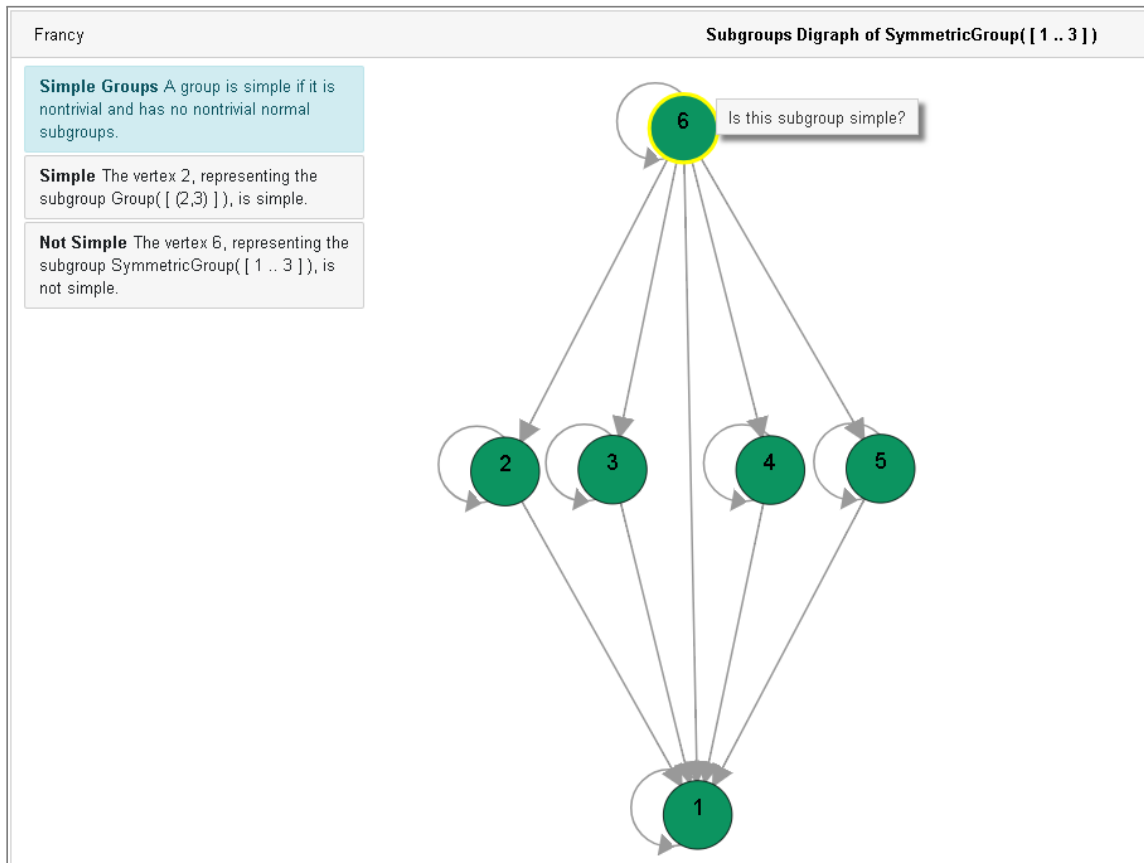


Figure 4.1: Directed graph of all subgroups of S_3

```

LoadPackage("digraphs"); LoadPackage("francy");

G:=SymmetricGroup(3); as:=AllSubgroups(G); nodes:=[];
d:=Digraph(as, {H, K} -> IsSubgroup(H, K));
vertices:=DigraphVertices(d); edges:=DigraphEdges(d);

canvas:=Canvas(Concatenation("Subgroups Digraph of ", String(G)));
graph:=Graph(GraphType.DIRECTED); Add(canvas, graph);

customMessage:=FrancyMessage(FrancyMessageType.INFO, "Simple
↪ Groups", "A group is simple if it is nontrivial and has no
↪ nontrivial normal subgroups.");

IsGroupSimple:=function(i)
  Add(canvas, customMessage);
  if IsSimpleGroup(as[i]) then
    Add(canvas, FrancyMessage("Simple", Concatenation("The vertex ",
↪ String(i), ", representing the subgroup ", String(as[i]),
↪ ", is simple.")));
  else
    Add(canvas, FrancyMessage("Not Simple", Concatenation("The vertex
↪ ", String(i), ", representing the subgroup ", String(as[i]),
↪ ", is not simple.")));
  fi;
  return Draw(canvas);
end;

for i in vertices do
  nodes[i]:=Shape(ShapeType.CIRCLE, String(i));
  Add(nodes[i], Menu("Is this subgroup simple?",
↪ Callback(IsGroupSimple, [i])));
  Add(graph, nodes[i]);
od;

for i in edges do
  Add(graph, Link(nodes[i[1]], nodes[i[2]]));
od;

Draw(canvas);

```

Listing 4.1: GAP code to display directed graph of all subgroups of S_3

Technical contribution

In terms of software design, *Francy* follows some principles such as Separation of Concerns and Modularity. These principles are perfectly articulated in the Computer Science Handbook [51] “*Any domain or application can be divided and decomposed into major building blocks and components (separation of concerns). This decomposition allows the application requirements to be further defined and refined, while partitioning these requirements into a set of interacting components (modularity). Changes to the application are (it is hoped) localised. In addition, team-oriented design and development can proceed with different team members concentrating on particular components*”.

Francy consists of two main components, a GAP package that is responsible for the semantic representation of graphics, and a second component, a GUI library that is responsible for generating the actual interactive graphical representation.

The GAP package creates a semantic representation of graphics, providing a thin layer between GAP objects and graphical objects to be rendered. This is done using JSON, a lightweight, text-based, language-independent data interchange format [34]. The semantic model follows a JSON Schema [52, 53], and is identified with the *application/vnd.francy+json* MIME type [21]. This creates an abstraction and allows the development of new GUI libraries, using different data rendering dependencies or even different programming languages, independently of the GAP package. This package is somehow based in XGAP throughout its application programming interface (API), but avoiding any non-GAP code. This has been the main concern, in order to allow a smooth integration with other GAP packages. In fact, *Francy* has only one dependency, the JSON package [54], that is distributed with GAP by default, and it is needed to communicate with Jupyter. Access to the GAP language shell (Read–Eval–Print Loop or REPL) is abstracted and managed by a kernel [55, 20].

At the moment, *Francy* has a JavaScript GUI library, based on d3.js [24], for rendering the semantic representation produced by the GAP package. This library is distributed both as a browser module and as a Jupyter extension. The browser module can be used for displaying graphics outside a Jupyter environment or to build applications that can be integrated with GAP, for instance, using web-sockets [56] and a web-based terminal emulator such as tty.js [57]. The Jupyter extension can be used in Jupyter Notebooks or Jupyter Lab, using the Jupyter GAP Kernel [20] and the MIME type *application/vnd.francy+json* to render the document.

Future work

Many other interactive features can be implemented providing a richer learning environment. Features such as rendering multiple topological graphs on the same canvas would allow, for instance, easier comparison of data structures. Other ways for users to input data would provide a more intuitive user experience.

Packages such as the *Francy-Monoids* [58], *Subgroup Lattice* [59] and the *Interactive Todd-Coxeter* [60] still need to be polished and finished.

At the moment, the semantic model based is not being validated against the JSON Schema [52]. This can be addressed in the future by extending the actual JSON package and implement the JSON Schema specification for validation of documents.

It would also be beneficial moving some of the processing JavaScript code into Web Workers [61], such that rendering of huge structures does not block the web page.

In some cases, having a local installation of *Francy* could be a requirement, and porting it to a desktop application is also possible as there are many tools to help on this process, for instance with ElectronJS [43].

4.2 XGAP vs Francy

The most significant difference between *Francy* and XGAP relies on the way objects are rendered and displayed to the user. XGAP is integrated with the X-Window system through a tightly coupled interface embedded in GAP, and the graphics are produced synchronously. This means that, e.g., nodes are rendered in the graphical canvas as soon they are produced in GAP. Here, all graphic objects are in sync and store their state in with GAP. Within *Francy*, this behaviour is not present. *Francy* is loosely coupled from GAP and the graphics are re-rendered on every call, meaning that, e.g., nodes are only rendered in the graphical canvas once the function *Draw* is called explicitly. Here graphical objects are not present in GAP and are instead referenced only by a *Francy* internal id. This makes all communication detached and asynchronous between the client and the server.

On the other hand, as mentioned before, *Francy* API is based on XGAP API, which will help users that are familiar with XGAP migrate to *Francy*. *Francys* API allows a seamless integration with any new or existing GAP package.

4.3 Francy Work Environment

Francy uses simple drawing and topological graph concepts, and allows the creation of charts or graphs on a canvas where it is possible to interact with simple shapes by clicking, zooming and spanning. These concepts are better explained later in this chapter.

Francy draws the graphics inside a canvas. This canvas contains an area for displaying messages an area for displaying the main menu and an area for the graphics. The main menu contains 2 menus by default, *Francy* and *Settings*.

The *Francy* menu has 3 sub-menus:

- Zoom to fit - fits the graph within the visible canvas area.
- Save to PNG - saves the visible canvas to a PNG image file.

CHAPTER 4. A GRAPHICS API FOR GAP - FRANCY

- About - shows information regarding Francy.

The *Settings* menu has 2 fixed sub-menus:

- Renderers - it is possible to select the JavaScript rendering engine to process the data, either a custom D3 implementation or a Graphviz engine or a Vis.js Network.
- Graph - only available when rendering graphs. Shows behavioural options for interaction with the graph.

Other options are available depending on the Renderer. If the selected Renderer is Graphviz, then the *Settings* menu will display:

- Viz Engine - the Graphviz engine to use, one of "circo", "dot", "fdp", "neato", "osage", "patchwork" or "twopi".
- Viz RankDir - the direction from which GraphViz will produce the graphics.

If the selected Renderer is Vis.js, then the *Settings* menu will display:

- Sort Method - "hubsize" or "directed", hubsize means the node with the most edges connected to it is on top. Alternatively, direction arranges the nodes based on the direction of the edges.
- Direction - the direction from which Vis.js will produce the graphics.

4.4 Francy Core API

The Core API is very simple and intuitive and is intended to be used by anyone who wants to integrate *francy* with an existing or new GAP package. The main functions or objects to retain are:

- Canvas - is the main object where graphs and charts are drawn in. It supports, at the moment, one chart or one graph.
- Graph - this object represents a topological graph. Within this object one can choose the type of graph to be drawn - tree, directed or undirected. These can be added to the object Canvas.
- Shape - this object represents the nodes in a topological graph and can be added to the object Graph.
- Chart - this object represents a chart, e.g. line chart, bar chart, etc. These can be added to the object Canvas.
- Callback - a callback is the mean to interact back with GAP and execute code. A callback has a trigger mechanism that can be - click, double click or right click. It can be added to Menu or Shape objects.

CHAPTER 4. A GRAPHICS API FOR GAP - FRANCY

- Menu - this function allows the creation of menus. These can be added to other Menu or to Shape objects.
- Add - this function references two or more objects together, e.g. *Add(canvas, graph)*; to add a graph to a canvas.
- FrancyMessage - this object stores a message to show to the user.
- Draw - this function returns a record containing a MIME type and the JSON data to be rendered. Within the right environment graphics are produced when this function returns.

The code snippet 4.2 shows a simple example of a graph containing one simple square shape that creates other circle shapes when it gets clicked. This code snippet produces the environment shown in figure 4.2.

```
canvas := Canvas("My First Canvas");
graph := Graph(GraphType.UNDIRECTED);
shape := Shape(ShapeType.SQUARE);
MyFunction := function()
  Add(graph, Shape(ShapeType.CIRCLE));
  return Draw(canvas);
end;
callback := Callback(TriggerType.CLICK, MyFunction);
Add(shape, callback);
Add(graph, shape);
Add(canvas, graph);
Draw(canvas);
```

Listing 4.2: A simple example of an interactive graph

CHAPTER 4. A GRAPHICS API FOR GAP - FRANCY

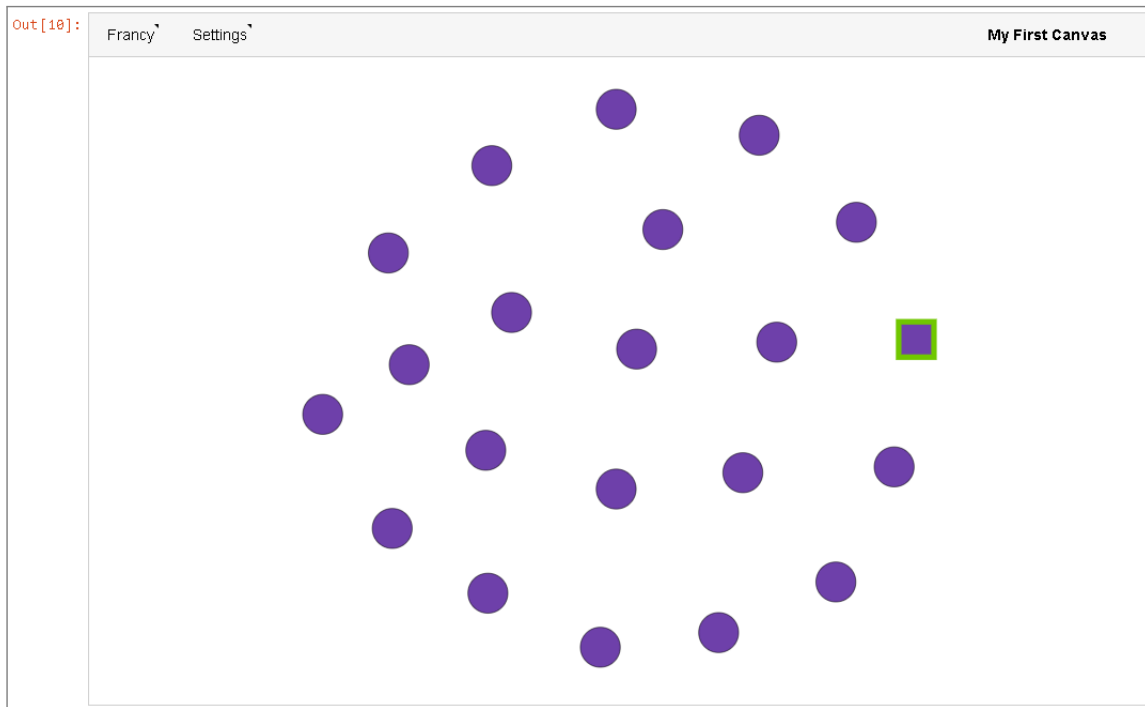


Figure 4.2: A simple example of an interactive graph

The complete *Francy* user manual can be seen in Appendix A on page 55.

Chapter 5

Packages using Francy

This chapter contains published results part of the regular GAP distribution.

This chapter introduces some examples of software packages using Francy.

5.1 Francy Monoids

`FrancyMonoids` is a GAP package that gives graphical representations to commutative monoids. By definition, a monoid is a set M equipped with a binary operation that is associative and has an identity element. If this binary operation is also commutative, i.e., if $ab = ba$ for every two elements a and b in M , then M is said to be a commutative monoid.

`FrancyMonoids` makes use of most of the functionality provided by Francy, and shows how simple it is to create a learning environment where students can interact and explore numerical semigroups. Having such a visual interactive environment helps understanding the theoretical concepts behind theorems.

This section presents unpublished results produced by Pedro A. García-Sánchez from Universidad de Granada, Andres Herrera-Poyatos from University of Oxford and myself. This research is based on published work produced by Pedro A. García-Sánchez, et al. [62, 63, 64, 65]

Procedures for elements in an affine semigroup

Let \mathbb{N} be the set of non negative integers, and let A be a subset of \mathbb{N}^r . The *monoid generated by A* is the set

$$\langle A \rangle = \{n_1 a_1 + \cdots + a_k a_k : k \in \mathbb{N}, n_i \in \mathbb{N}, a_i \in A, i \in \{1, \dots, k\}\}.$$

A submonoid S of $(\mathbb{N}^r, +)$ is an *affine semigroup* if there exists a finite $A \subset S$ with $\langle A \rangle = S$. We then say that A is a *generating system* of S . A generating system is minimal if there is no proper subset of this system of generators that generates S .

CHAPTER 5. PACKAGES USING FRANCY

Every affine semigroup admits a unique minimal generating system; its elements are called the *minimal generators* of S .

We say that S is a *numerical semigroup* if S is a submonoid of $(\mathbb{N}, +)$ with finite complement in \mathbb{N} . Every numerical semigroup is finitely generated, and thus numerical semigroups are affine semigroups.

Let $A = \{a_1, \dots, a_e\} \subset \mathbb{N}^r$, with e a positive integer. We say that (z_1, \dots, z_e) is a *factorization* of $a \in \langle A \rangle$ if $a = z_1 a_1 + \dots + z_e a_e$. The *length* of a factorization z is $|z| = z_1 + \dots + z_e$.

Given two factorizations z and z' of a , we define

$$z \wedge z' = (\min(z_1, z'_1), \dots, \min(z_e, z'_e)),$$

and the *distance* between z and z' as

$$d(z, z') = \max(|z - (z \wedge z')|, |z' - (z \wedge z')|).$$

An *N-chain* of factorizations of a joining z and z' is a sequence of factorizations z_1, \dots, z_t of a such that $z_1 = z$, $z_t = z'$ and $d(z_i, z_{i+1}) \leq N$ for all i . The *catenary degree* of a is the least integer c such that for any two factorizations of a there is a c -chain of factorizations joining them.

Factorizations are governed by a minimal presentation of the affine semigroup $\langle A \rangle$. Minimal presentations can either be obtained by looking at the associated Eliahou or Rosales graphs associated to some specific elements in the semigroup (those having nonnconected graphs, and the relations are obtained by looking at the nonnconected components).

DrawFactorizationGraph(IsRectangularTable)

DrawFactorizationGraph(f) - Returns: a drawing

Given a set of factorizations f , this function draws the graph of factorizations associated to f : a complete graph whose vertices are the elements of f . Edges are labelled with distances between nodes they join. Kruskal algorithm is used to draw in red a spanning tree with minimal distances. Thus the catenary degree is reached in the edges of the tree.

For instance, code listing 5.1 shows the GAP code for $S = \langle 3, 5, 7 \rangle$ and $30 \in S$, and the result set of factorizations is shown in code listing 5.2.

```
gap> s:=NumericalSemigroup(3,5,7);
gap> f:=FactorizationsElementWRTNumericalSemigroup(30,s);
```

Listing 5.1: Create a numerical semigroup and factorizations

```
gap> f;
[ [ 10, 0, 0 ], [ 5, 3, 0 ], [ 0, 6, 0 ], [ 6, 1, 1 ], [ 1, 4, 1
  ↪ ], [ 2, 2, 2 ], [ 3, 0, 3 ] ]
```

Listing 5.2: Factorizations of numerical semigroup result

The output of `DrawFactorizationGraph(f)`, using Graphviz-Renderer with *circ* engine, is shown in figure 5.1.

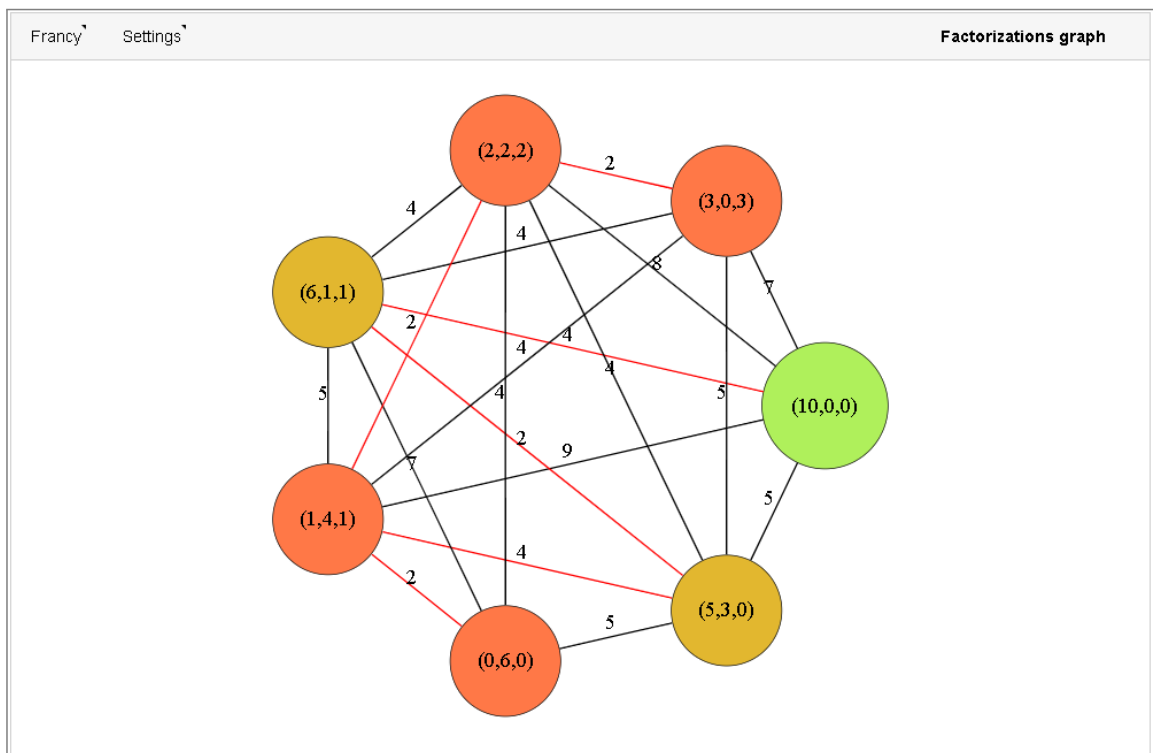


Figure 5.1: Factorizations graph using Graphviz-Renderer with circ engine

DrawEliahouGraph (for IsRectangularTable)

`DrawEliahouGraph(f)` - Returns: a drawing

Given a set of factorizations f , this function draws the Eliahou graph of factorizations associated to f : a graph whose vertices are the elements of f , and there is an edge between two vertices if they have common support (equivalently, their dot product is nonzero). Edges are labelled with distances between nodes they join.

The output of `DrawEliahouGraph(f)` is shown in figure 5.2.

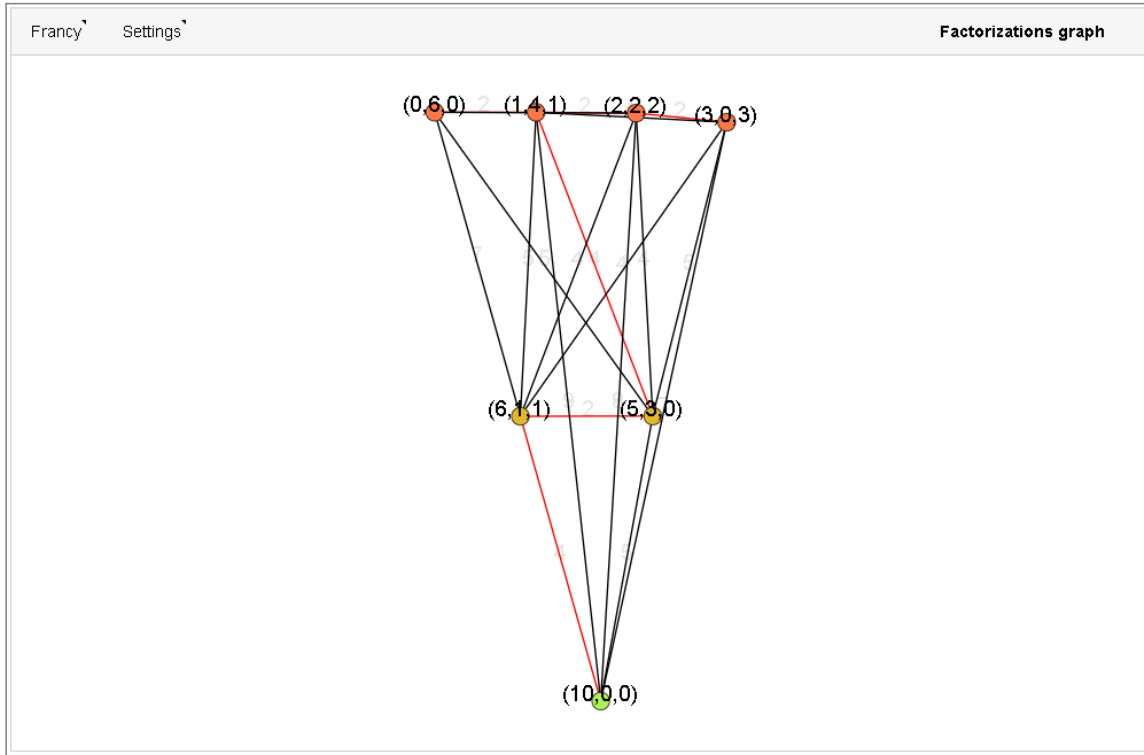


Figure 5.2: Factorizations Eliahou Graph using D3 Renderer

DrawRosalesGraph(IsHomogeneousList, IsAffineSemigroup)

DrawRosalesGraph(n, S) - Returns: a drawing

Given S is either a numerical semigroup or an affine semigroup, and n is an element of S , this function draws the graph associated to n in S . This graph has as vertices the set of minimal generators a of S with $n - a \in S$, and ab is an edge provided that $n - (a + b) \in S$.

The output of DrawRosalesGraph($10, s$) is shown in figure 5.3.

Numerical semigroup specific functions

Let S be a numerical semigroup. Since $\mathbb{N} \setminus S$ has finitely many elements, the set of oversemigroups of S , $\{T : T \text{ numerical semigroup}, S \subseteq T\}$, has finitely many elements.

DrawOverSemigroupsNumericalSemigroup

DrawOverSemigroupsNumericalSemigroup(s) - Returns: a drawing

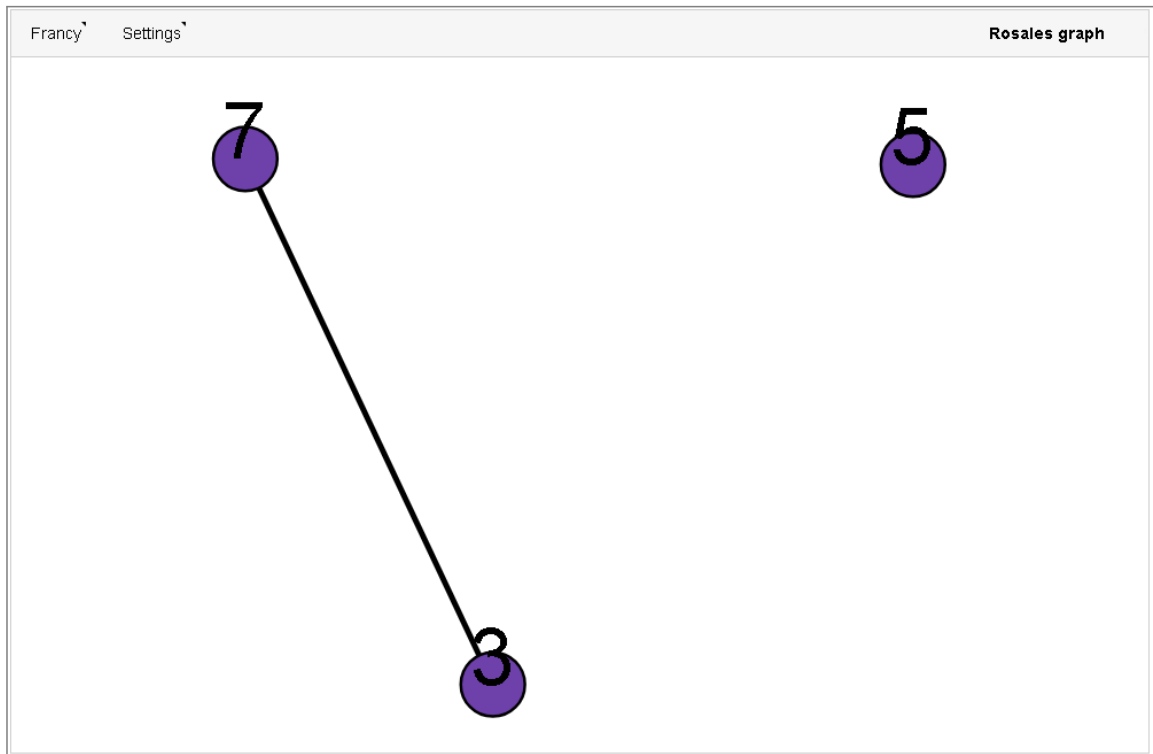


Figure 5.3: Rosales Graph using D3 Renderer

Draws the Hasse diagram of oversemigroupstree of the numerical semigroup s . Irreducible numerical semigroups are highlighted.

So for instance, the code listing 5.3 produces the graph shown in figure 5.4.

```
gap> s:=NumericalSemigroup(5,7,11,13);
gap> DrawOverSemigroupsNumericalSemigroup(s);
```

Listing 5.3: Create a numerical semigroup and draw the over semigroups

In figure 5.4, rhomboid nodes correspond to irreducible numerical semigroups, that is, those numerical semigroups that cannot be expressed as the intersection of two other semigroups properly containing them. By looking at this picture, one easily sees the ways $\langle 5, 7, 11, 13 \rangle$ decomposes as the intersection of irreducible numerical semigroups.

DrawTreeOfSonsOfNumericalSemigroup

`DrawTreeOfSonsOfNumericalSemigroup($s, l, generators$)` - Returns: a drawing

Draws the tree of sons of numerical semigroups up to level l with respect to the minimal system of generators given by the function `generators`.

Given a numerical semigroup S and $a \in S$, the set $S \setminus \{a\}$ is a numerical semigroup if and only if a is a minimal generator. If this is the case we say that $S \setminus \{a\}$ is a *son* of S or a *descendant*. The process can be repeated up to the desired level.

CHAPTER 5. PACKAGES USING FRANCY

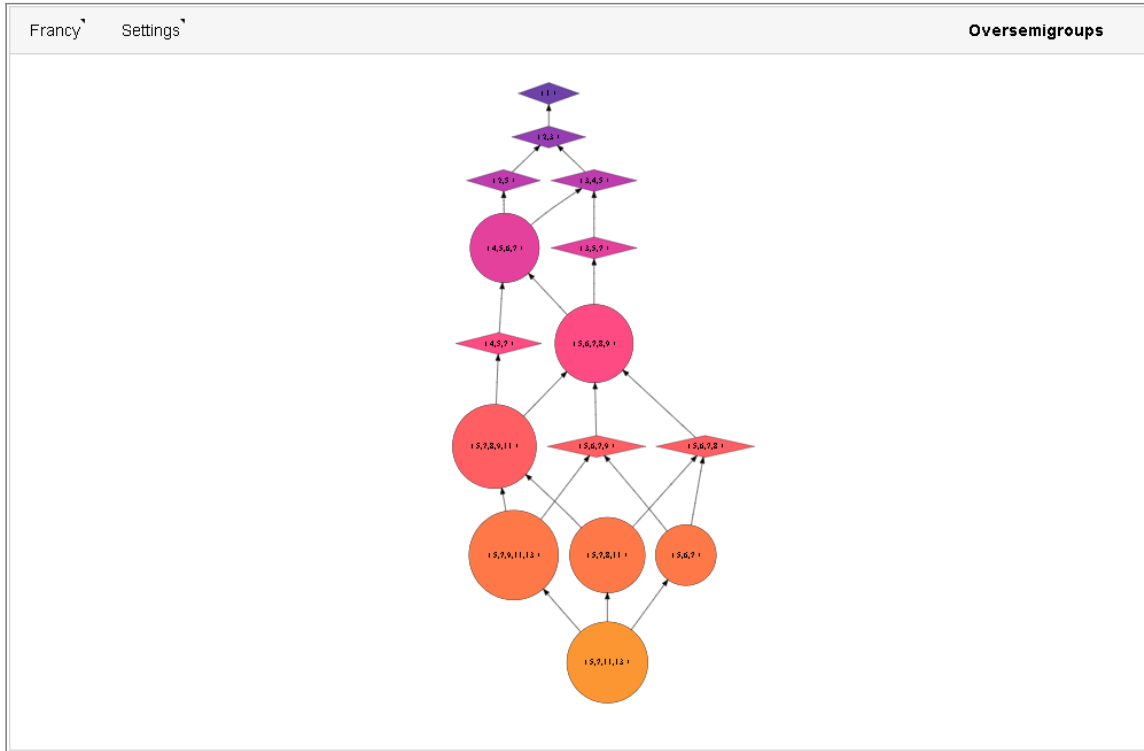


Figure 5.4: Hasse diagram of oversemigroups using Graphviz Renderer

The growth in depth is known to be like Fibonacci sequence, and thus one cannot expect to get down to a very high level.

An example of such tree is shown in figure 5.5

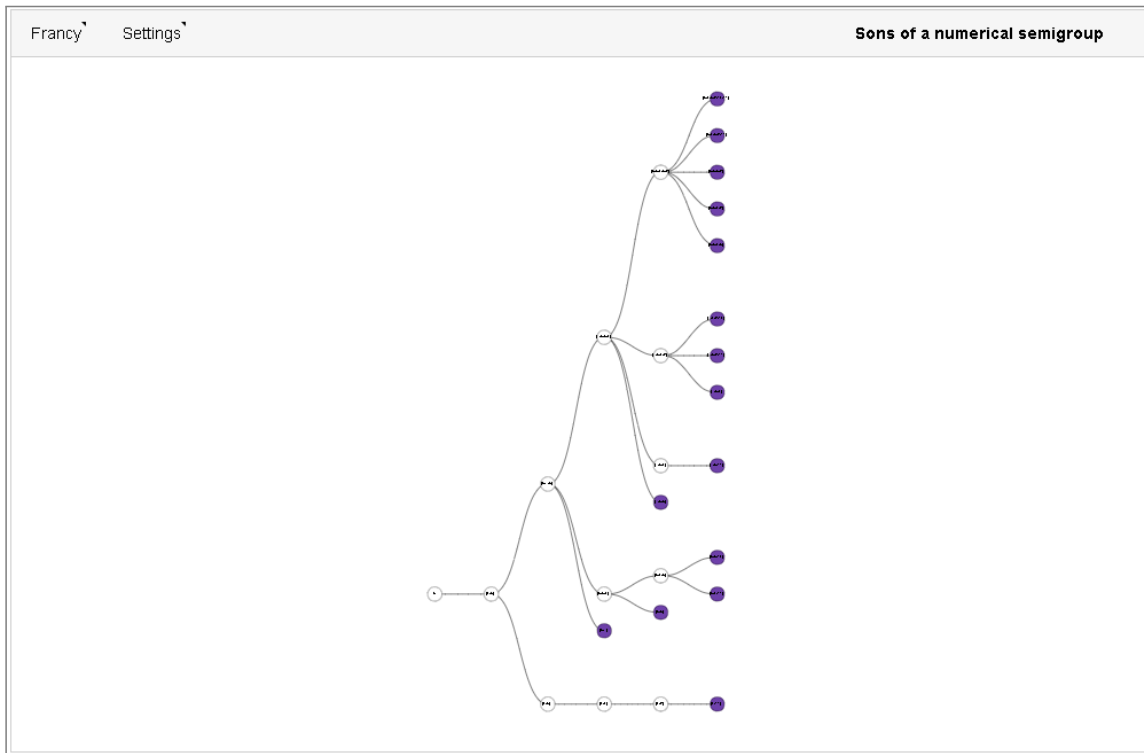


Figure 5.5: Tree of sons of numerical semigroups using D3 Renderer

DrawHasseDiagramOfNumericalSemigroup

`DrawHasseDiagramOfNumericalSemigroup(S, A)` - Returns: a drawing

Plots a graph whose set of vertices is A , which is a finite set of integers, and whose edges are provided by the order of the numerical semigroup S , that is, $a \leq b$ if $b - a \in S$.

So for instance, the output for $S = \langle 3, 5, 7 \rangle$ and $A = \{1, \dots, 20\}$ is shown in figure 5.6.

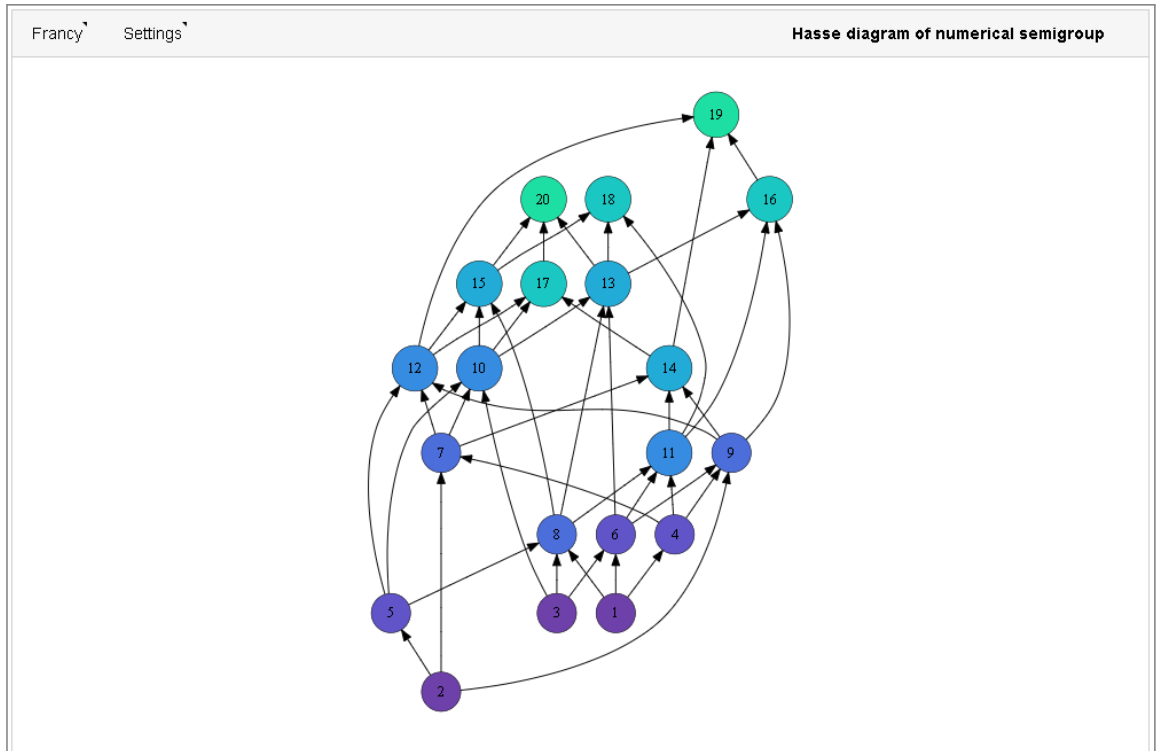


Figure 5.6: Hasse Diagram using Graphviz Renderer

DrawTreeOfGluingOfNumericalSemigroup

`DrawTreeOfGluingOfNumericalSemigroup($S[, \text{expand}]$)` - Returns: a drawing

Returns a Francy canvas with the tree of gluings of the numerical semigroup S . If the optional argument `expand` is provided, then the tree is drawn fully expanded.

We say that $S = a_1 S_1 + a_2 S_2$ is a gluing provided that a_1 is an element in S_2 that is not a minimal generator of S_2 , a_2 is an element of S_1 and not a minimal generator, and $\gcd(a_1, a_2) = 1$. Some properties of S can be inferred from S_1 and S_2 (as minimal presentations, Frobenius number, ...).

Not every numerical semigroup is a gluing.

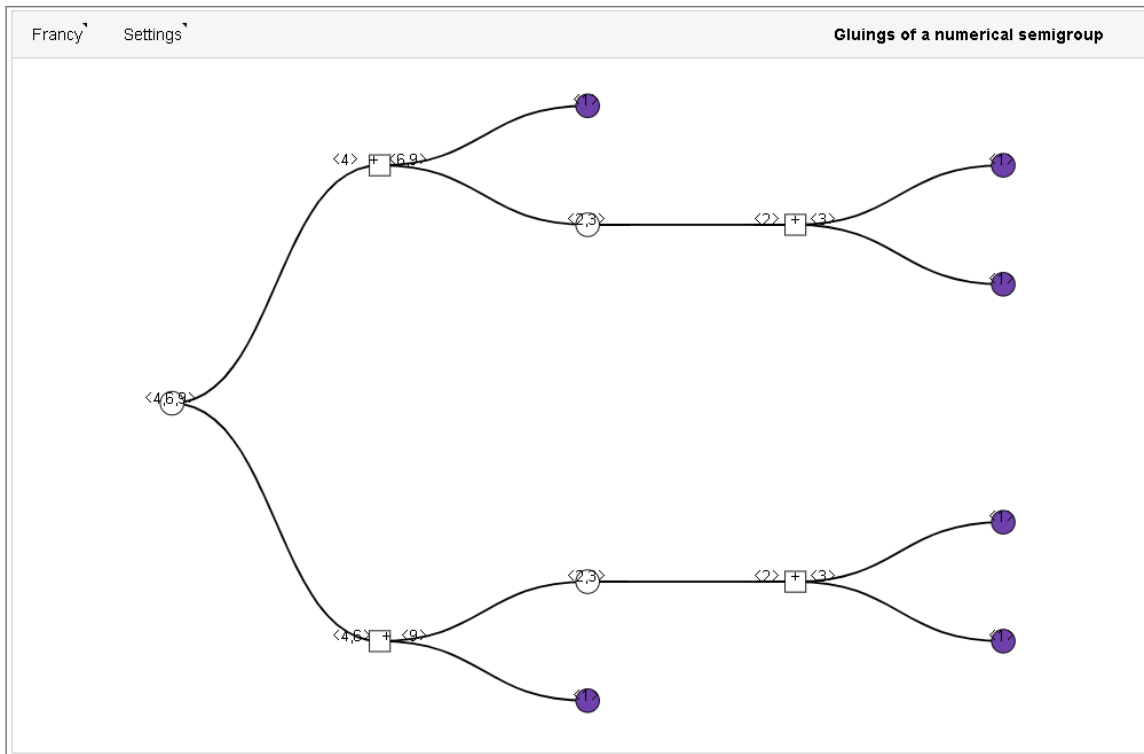


Figure 5.7: Tree of Gluing using D3 Renderer

5.2 Subgroup Lattice

The Subgroup Lattice software package for GAP is based on the XGAP algorithms and shows how Francy can be used to provide an interactive environment for exploring lattices.

This package, at the moment, provides only basic features. It is possible to display all subgroups lattice and get all the properties of the subgroups by accessing the context menu on right-clicking.

Features

Starting with the Symmetric Group of order 4, S_4 , and similarly to XGAP, by calling the function `GraphicSubgroupLattice(G)`; Francy will display the subgroup lattice in an interactive environment, by default the group G itself and the trivial subgroup 1. Figure 5.8 shows this representation using the D3 renderer.

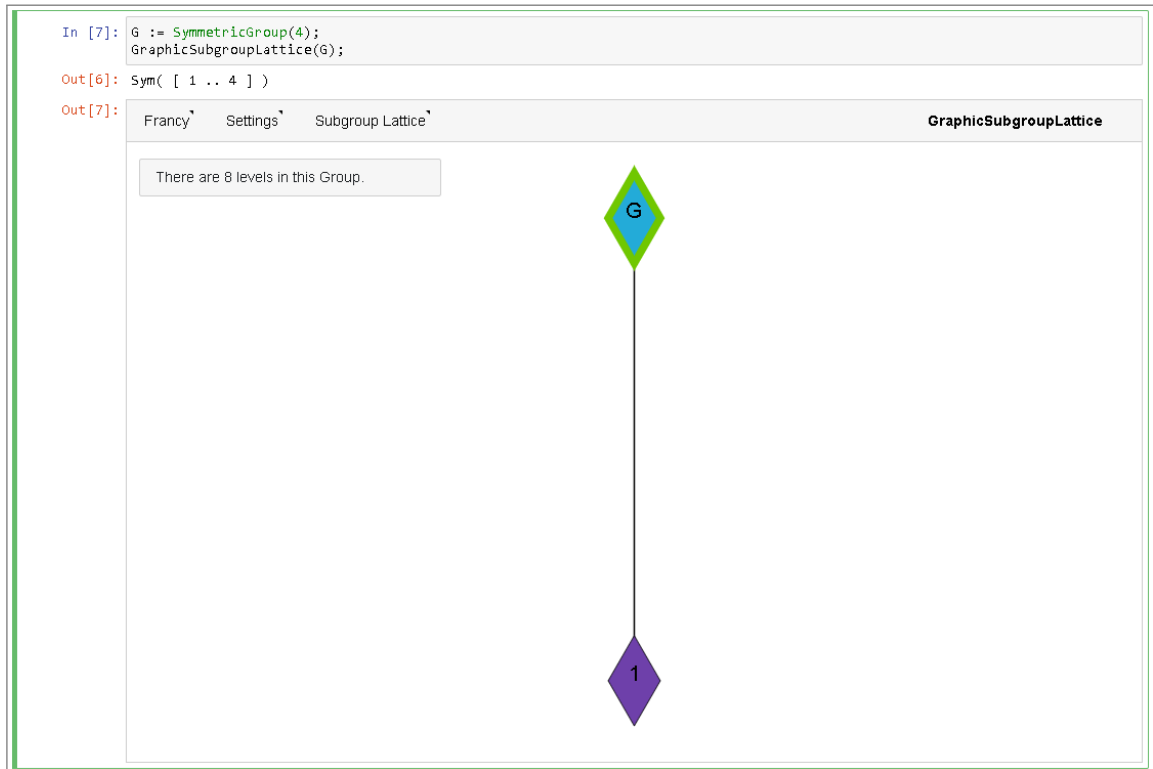


Figure 5.8: Subgroup lattice of S_4 , using D3 Renderer

CHAPTER 5. PACKAGES USING FRANCY

By interacting with Francy, in this specific example for the Group S_4 , using the main menu is possible to display all subgroups lattice by selecting *All Subgroups* in the *Subgroup Lattice* Menu. This will display the lattice subgroups in a Hasse Diagram. Francy will display each level on this diagram with a different colour representing the order of the subgroup. Figure 5.9 shows this representation using the D3 renderer.

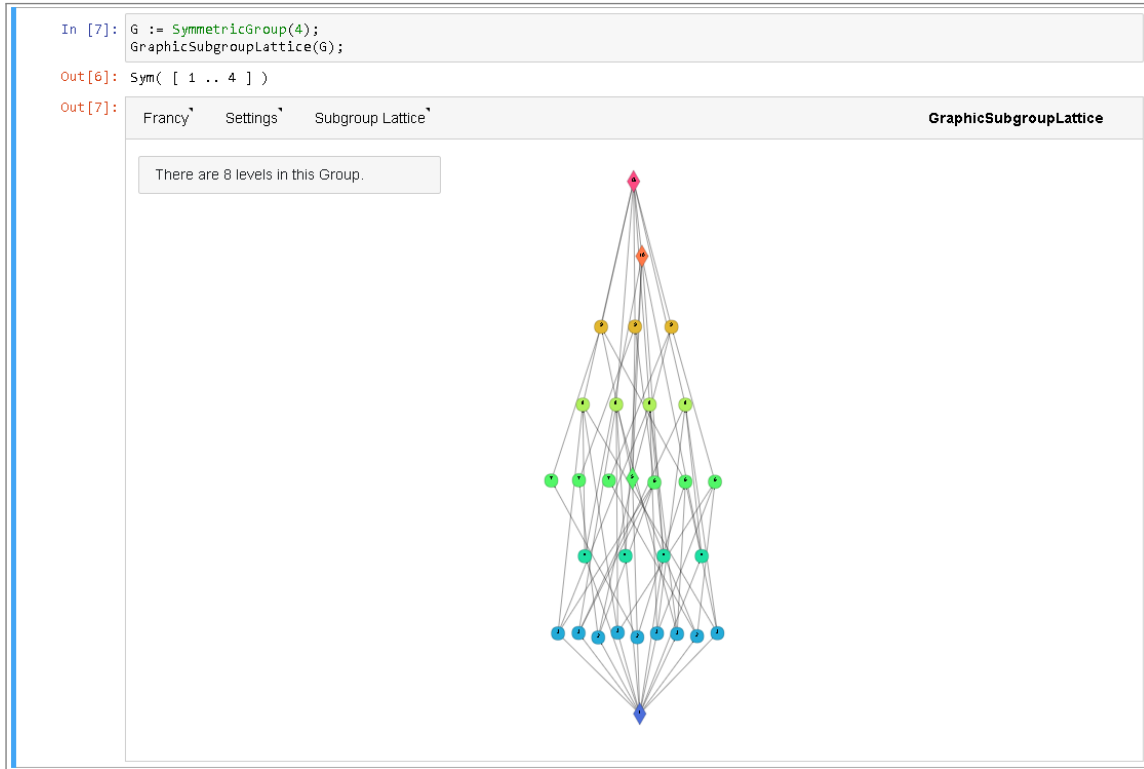


Figure 5.9: All subgroups lattice of S_4 , using D3 Renderer

In order to study the subgroups properties, by right clicking any subgroup, represented by the nodes in the different levels of the Hasse Diagram, a context menu will be displayed. Within this menu it is possible to click and get the property value for that specific subgroup. Figure 5.10 shows this functionality.

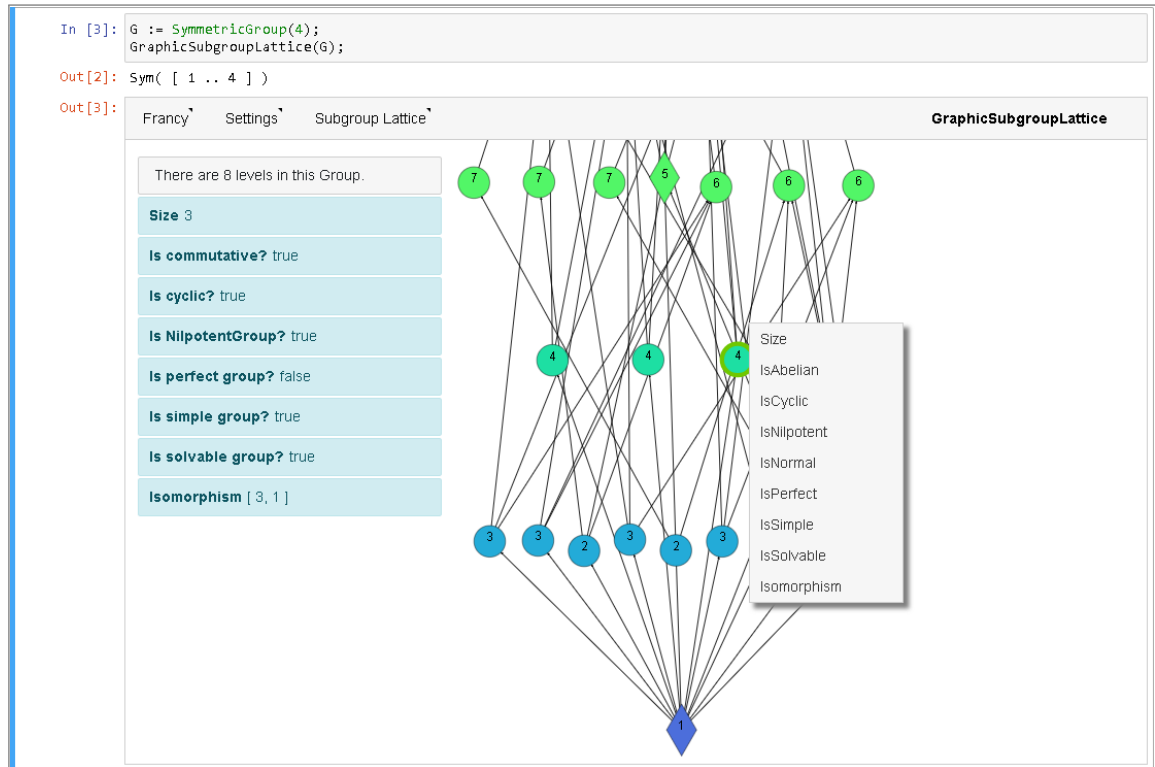


Figure 5.10: Subgroup lattice of S_4 , subgroup properties

Chapter 6

Conclusion

For this dissertation, I successfully implemented software projects capable of providing a better user experience to users of GAP.

The first, capable of running gap on the web - *WebGAP* - where users could collaborate in a rich environment. The second, to enhance usability of GAP - *GAP Lint* and *GAP CodeMirror Mode* - helping users writing and learning GAP code, improving code quality and efficiency. The third, to provide a Graphical User Interface for GAP - *Francy*, that can be used to enrich GAP packages, providing means to interact with mathematical objects.

WebGAP proved GAP could run on the web, without much effort and in a secure way by running on low privileges and within Docker containers which offer process isolation. This project introduced me to the GAP community and allowed me to contribute to their packages ecosystem.

The GAP highlighting and linting functionality help users learning and writing code enriching the experience of using GAP.

Francy is the first GUI framework for GAP on the web and allowing interactive packages on Jupyter.

The software development cycle is a never ending fine tuning process, and the release success depends mostly on finding the right compromise between maturity and features. Other external factors could influence it, and the author acknowledges that either *WebGAP* and *Francy* should have been released before. *WebGAP* could have been a stable platform that would, the same way as Jupyter, provide access to many other computational tools. In *Francy's* case, releasing some months before would have resulted in many more use cases for the *OpenDreamKit* project.

6.1 Contributions

My contributions to GAP can be described mainly in the areas of Graphical User Interface and in User Experience for GAP on the web. These can be seen not only

CHAPTER 6. CONCLUSION

as new packages or modules, but as new paradigms to extend GAP's functionality, using decoupling design patterns.

Francy brings new concepts to the GAP community and is the first graphical user interface that does not require any change to GAP's core, but uses instead a decoupled strategy.

The integration of *Francy* with Jupyter within the OpenDreamKit Project was a success. *Francy* is described as a breakthrough in terms of design and should be explored on future developments of mathematical software.

The integration of tools to enhance GAP usage, specially the code highlighting tool, within the GAP Jupyter Kernel, provide a better user experience overall, when writing GAP code within the Jupyter environment.

Interactive content for teaching purposes can now be produced and presented in a faster and simple way, providing other means to explain and show theoretical concepts in the form of visual graphics.

These contributions called attention of the OpenDreamKit community in general, and SageMath, Inc. one of the projects' partners, is willing to reuse all of them within their commercial product - CoCalc [66].

6.2 Limitations

There are some limitations in the software produced for this dissertation.

WebGAP started as simple web application providing access to applications running on Unix systems. Even if all security considerations were being taken, *WebGAP* would never been as robust and secure as an application developed, reviewed and tested by a large community of developers as Jupyter is. The fact that *WebGAP* never had a major release is mainly due to the huge effort that developing such a big framework takes, all the considerations that have to be taken about security, releasing and maintenance.

As *Francy* has been built having in mind the same concepts of *XGAP*, one should be conscious about the core differences between both, when building packages using it. The asynchronous communication with GAP forces the rendering of the whole graph within *Francy* environment. Some efforts to minimise this effect were implemented and the D3 renderer is capable of handling merges between calls to GAP, by not re-draw existing nodes making all the animations smoother. At the moment both the *GraphViz* and the *Vis.js* renderers do not handle these transitions smoothly, as they do not implement any merging strategy. When implementing new renderers for *Francy*, one has to have these considerations in mind.

6.3 Future Work

There is still a lot that can be done to enrich its functionality and ensure better tools for research and teaching.

As mentioned before, I think a formalisation of an official grammar for GAP would enable many more features to be developed helping users to engage with GAP development.

Francy still lacks some functionality. Some components, such as free drawing, e.g. points, text, lines, etc, are still lacking and are required to make sure some old GAP packages can be brought to the web, such as the Interactive Todd Coxeter package. Other improvements can be made to allow a smoother graphical experience, such as relying on HTML Canvas instead of SVG for graphics and use of Web Workers for rendering large datasets. All these ideas can be implemented as features and should not require any major changes to Francy's core.

The Subgroup Lattice package provides an interactive environment where one can study certain properties of any given Group. Other functionality, currently present on XGAP, can be migrated to this new package. Functionality such as *Selected Groups to GAP*, which takes the selected vertices and returns a list consisting of the associated subgroups of the group, or *Sylow Subgroups* which takes a prime and returns all the sylow subgroups for that same prime, are invaluable for exploring the subgroup lattice of large groups. Many other functionality can be added to enrich this environment.

Bibliography

- [1] Manuel Machado Martins. Webgap portal.
<https://bitbucket.org/mcmartins/webgap>, 2014.
- [2] Manuel Machado Martins. Webgap portal v2.
https://bitbucket.org/mcmartins/webgap_v2, 2015.
- [3] GAP Group. Gap - groups, algorithms, and programming, version 4.10.0.
<https://www.gap-system.org/>, 2018.
- [4] ahp. ahp - application and hosting provider.
<https://www.ahp.pt/>, 2019.
- [5] Community OpenDreamKit. Project opendreamkit.
<http://opendreamkit.org/>, 2018.
- [6] Community OpenDreamKit. Project opendreamkit.
<https://opendreamkit.org/about/>, 2018.
- [7] Community OpenDreamKit. Open digital research environment toolkit for the advancement of mathematics.
<https://github.com/OpenDreamKit/OpenDreamKit/raw/master/Proposal/proposal-www.pdf>, 2017.
- [8] Sebastian Gutsche Jeroen Demeyer and Nicolas M. Thiéry. Report on opendreamkit deliverable d4.7 - full featured jupyter interface for gap, pari/gp, singular.
<https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP4/D4.7/report-final.pdf>, 2018.
- [9] Odile Bénassy and Nicolas M. Thiéry. Report on opendreamkit deliverable d4.16 - exploratory support for semantic-aware interactive widgets providing views on objects represented and or in databases.
<https://github.com/OpenDreamKit/OpenDreamKit/blob/master/WP4/D4.16/report-final.pdf>, 2018.
- [10] N. Carter. Jupyter-Viz, jupyter notebook visualization tools, Version 1.1.1.
<https://nathancarter.github.io/jupyterviz>, Oct 2018. GAP package.
- [11] García-Sánchez. Github.
<https://github.com/pedritomelenas>, 2018.

BIBLIOGRAPHY

- [12] Russ Woodroffe. Introducing gap.app.
<https://cocoagap.sourceforge.io/>, 2018.
- [13] Terence Parr. *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, 2nd edition, 2013.
- [14] Microsoft. Microsoft azure for research.
<https://www.microsoft.com/en-us/research/academic-program/microsoft-azure-for-research/>, 2018.
- [15] Markus Pfeiffer. Cloudgap - jupyter.
<https://cloudgap.gap-system.org/>, 2018.
- [16] Frank Celler and Max Neunhöffer. Xgap documentation, what is xgap?
<https://www.gap-system.org/Manuals/pkg/xgap-4.26/htm/CHAP002.htm>, 2018.
- [17] Community Jupyter. Project jupyter.
<http://jupyter.org/>, 2018.
- [18] Community IPython_Contributors. Unofficial jupyter notebook extensions.
<https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/>, 2018.
- [19] Community Jupyter. Jupyter documentation, jupyter kernels.
<https://jupyter-client.readthedocs.io/en/stable/kernels.html>, 2018.
- [20] M. Pfeiffer and M Martins.
JupyterKernel, jupyter kernel written in gap, Version 1.0.
<https://gap-packages.github.io/JupyterKernel/>, Oct 2018. GAP package.
- [21] Mozilla and individual contributors. Mime types.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types, 2018.
- [22] Marijn Haverbeke. Codemirror.
<https://codemirror.net/>, 2018.
- [23] M. Martins.
Francy, framework for interactive discrete mathematics, Version 1.0.3.
<https://gap-packages.github.io/francy>, Oct 2018. GAP package.
- [24] Mike Bostock. Data-driven documents, d3.
<https://d3js.org/>, 2018.
- [25] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen North, Gordon Woodhull, Short Description, and Lucent Technologies. Graphviz — open source graph drawing tools. In *Lecture Notes in Computer Science*, pages 483–484. Springer-Verlag, 2001.

BIBLIOGRAPHY

- [26] visjs community. Vis.js.
<https://visjs.org/>, 2019.
- [27] IBM Global Services. The power of cloud driving business model innovation, 2012.
http://www.ibm.com/midmarket/us/en/att/pdf/The_power_of_cloud_Executive_Report.pdf.
- [28] Rackspace Support. Understanding the cloud computing stack:saas paas iaas, 2013.
<https://support.rackspace.com/how-to/understanding-the-cloud-computing-stack-saas-paas-iaas/>.
- [29] Luis M. Vaquero, Luis Roderio-Merino, and Rajkumar Buyya. Dynamically scaling applications in the cloud. *SIGCOMM Comput. Commun. Rev.*, 41(1):45–52, January 2011.
- [30] Frank Buschmann. *Pattern-oriented software architecture. [Vol. 1], A system of patterns*. Chichester ; New York : Wiley, 1996. Includes bibliographical references (pages 441-453) and index.
- [31] James Coglan. Faye simple pub/sub for the web, internal architecture, 2013.
<http://faye.jcoglan.com/architecture.html>.
- [32] E.S. Raymond. *The Art of UNIX Programming*. Addison-Wesley Professional Computing Series. Pearson Education, 2003.
- [33] OAuth Working Group. Json web token (jwt), draft-ietf-oauth-json-web-token-23, 2014.
<http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html>.
- [34] Douglas Crockford. The javascript object notation (json) data interchange format.
<https://tools.ietf.org/html/rfc8259>, 2018.
- [35] Nicholas Rosasco and David Larochelle. *How and Why More Secure Technologies Succeed in Legacy Markets*, pages 247–254. Springer US, Boston, MA, 2004.
- [36] OpenID Foundation. Benefits of openid, 2014.
<http://openid.net/get-an-openid/individuals/>.
- [37] Pivotal. Rabbitmq.
<https://www.rabbitmq.com/>, 2018.
- [38] Krishna Srinivas. Terminal in browser over http/https. (ajaxterm/anyterm alternative, but much better), 2014.
<https://github.com/krishnasrinivas/wetty>.
- [39] Docker inc. Docker.
<https://www.docker.com/>, 2018.

BIBLIOGRAPHY

- [40] Adobe / Brackets Community. Brackets ide web page.
<http://brackets.io/>, 2018.
- [41] Amazon / Cloud9 Community. Cloud9 github.
<https://github.com/c9>, 2018.
- [42] GitHub. Github.
<https://ide.atom.io/>, 2019.
- [43] Electron Community. Build cross platform desktop apps with javascript, html, and css.
<https://electronjs.org/>, 2018.
- [44] Manuel Machado Martins. Gap linter github page.
<https://github.com/mcmartins/gap-lint>, 2018.
- [45] ANTLR Community. Antlr4 github.
<https://github.com/antlr/antlr4>, 2018.
- [46] Bnf notation syntax.
<https://www.w3.org/Notation.html>, 2018.
- [47] Manuel Machado Martins and Markus Pfeiffer. Francy - an interactive discrete mathematics framework for gap. In James H. Davenport, Manuel Kauers, George Labahn, and Josef Urban, editors, *Mathematical Software – ICMS 2018*, pages 352–358, Cham, 2018. Springer International Publishing.
- [48] J. Berg. *Visual Leap: A Step-by-Step Guide to Visual Learning for Teachers and Students*. Bibliomotion, Incorporated, 2015.
- [49] Ludger Hippe Volkmar Felsch and Joachim Neubüser. "ITC documentation", what is ITC?
<https://www.gap-system.org/Manuals/pkg/itc/htm/CHAP001.htm>, 2018.
- [50] Jan De Beule, Julius Jonušas, James D. Mitchell, Michael Torpey, and Wilf A. Wilson. Digraphs - gap package, version 0.12.1, Apr 2018. doi: 10.5281/zenodo.596465.
- [51] A.B. Tucker. *Computer Science Handbook, Second Edition*. CRC Press, 2004.
- [52] Manuel Machado Martins. Francy schema github page.
<https://github.com/mcmartins/francy/blob/master/schema/francy.json>, 2018.
- [53] Community Json_Schema. Json schema.
<http://json-schema.org/>, 2018.
- [54] Christopher Jefferson. json - reading and writing json.
<https://www.gap-system.org/Manuals/pkg/json-1.2.0/doc/chap0.html>, 2018.

BIBLIOGRAPHY

- [55] Jupyter Community. Jupyter documentation, jupyter kernel gateway.
<http://jupyter-kernel-gateway.readthedocs.io/en/latest/>, 2018.
- [56] Mozilla and individual contributors. Websockets.
https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API, 2018.
- [57] Christopher Jeffrey. tty.js github page.
<https://github.com/chjj/tty.js>, 2018.
- [58] P. A. García-Sánchez, A. Herrera-Poyatos, and M. Martins. FrancyMonoids, francymonoids/a package to display commutative monoid objects with francy, Version 0.1.
<https://gap-packages.github.io/FrancyMonoids>, Jun 2018. GAP package.
- [59] Manuel Machado Martins. Subgroup lattice github page.
<https://github.com/mcmartins/subgroup-lattice>, 2018.
- [60] Manuel Machado Martins. Interactive todd-coxeter github page.
<https://github.com/mcmartins/interactive-todd-coxeter>, 2018.
- [61] Mozilla and individual contributors. Web workers api.
https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API, 2018.
- [62] P. A. García-Sánchez. An overview of the computational aspects of nonunique factorization invariants. In Scott Chapman, Marco Fontana, Alfred Geroldinger, and Bruce Olberding, editors, *Multiplicative Ideal Theory and Factorization Theory*, pages 159–181, Cham, 2016. Springer International Publishing.
- [63] J.C. Rosales and P.A. García-Sánchez. *Finitely Generated Commutative Monoids*. Nova Science Publishers, 1999.
- [64] J.C. Rosales and P.A. García-Sánchez. *Numerical Semigroups*. Developments in Mathematics. Springer New York, 2009.
- [65] A. Assi and P.A. García-Sánchez. *Numerical Semigroups and Applications*. RSME Springer Series. Springer International Publishing, 2016.
- [66] Inc SageMath. Cocalc - collaborative calculation in the cloud.
<https://cocalc.com/>, 2018.

Appendices

Appendix A

Francy User Manual

This section presents, *as is*, the documentation produced for the GAP package Francy.

The documentation for the latest version of the package can be found on the GAP website - <https://www.gap-system.org/Packages/francy.html>

Introduction

Francy

Francy arose from the necessity of having a lightweight framework for building interactive graphics, generated from GAP, running primarily on the web, primarily in a Jupyter Notebook. An initial attempt to re-use XGAP and port it was made, but the lack of a standardised data exchange format between GAP and the graphics renderer, and the simplistic initial requirements of the project were the basis for the creation of a new GAP package.

Applications

Francy has potentially many applications and can be used to provide a graphical representation of data structures, allowing one to navigate through and explore properties or relations of these structures. In this way, Francy can be used to enrich a learning environment where GAP provides a library of thousands of functions implementing algebraic algorithms as well as large data libraries of algebraic objects. Example packages using Francy: FrancyMonoids, SubgroupLattice

Functionality

Francy provides an interface to draw graphics using objects. This interface is based on simple concepts of drawing and graph theory, allowing the creation of directed and undirected graphs, trees, line charts, bar charts and scatter charts. These graphical objects are drawn inside a canvas that includes a space for menus and to display informative messages. Within the canvas it is possible to interact with the graphical objects by clicking, selecting, dragging and zooming

Installation

This package requires the GAP packages Jupyter Kernel and json, all of which are distributed with GAP. Francy follows a similar installation procedure to Jupyter Kernel, so it requires Jupyter to be installed on your system. Please note, you need to run the installation command from the same python version Jupyter is installed on. In order to install Francy, please run the following command to download the latest version available from <https://pypi.org/>:

```
$ pip install jupyter_francy -U
```

Listing A.1: Jupyter extension installation

It is then necessary to enable Francy on your Jupyter installation:

```
$ jupyter nbextension enable --py --sys-prefix jupyter_francy
```

Listing A.2: Jupyter extension enable

How it works

Francy requires a rendering package to display graphics. Francy uses Renderers based on d3.js, Graphviz and Vis.js, for rendering the semantic representation produced by the GAP package. This library is distributed both as a browser module and as a Jupyter extension. The Jupyter extension can be used in Jupyter Notebooks or Jupyter Lab, using the Jupyter Kernel and the MIME type application/vnd.francy+json to render the document. Please check the documentation for more information: [JavaScript Documentation](#).

Francy Callbacks

Callbacks are objects that hold a function, a list of arguments and a trigger event. Callbacks are used to execute GAP code from a remote client using the Trigger Operation.

Callbacks can be added directly to Menus and/or Shapes.

Please see Francy-JS for client implementation.

Categories

In this section we show the Francy Callback Categories.

IsCallback

Identifies Callback objects.

IsCallback(arg) - Returns: true or false

IsRequiredArg

Identifies RequiredArg objects.

IsRequiredArg(arg) - Returns: true or false

IsArgType

Identifies ArgType objects.

IsArgType(arg) - Returns: true or false

IsTriggerType

Identifies TriggerType objects.

IsTriggerType(arg) - Returns: true or false

Representations

In this section we show the Francy Callback Representations.

IsCallbackRep

Checks whether an Object has a Callback internal representation.

IsCallbackRep(arg) - Returns: true or false

IsRequiredArgRep

Checks whether an Object has a RequiredArg internal representation.

IsRequiredArgRep(arg) - Returns: true or false

IsArgTypeRep

Checks whether an Object has a ArgType internal representation.

IsArgTypeRep(arg) - Returns: true or false

IsTriggerTypeRep

Checks whether an Object has a TriggerType internal representation.

IsTriggerTypeRep(arg) - Returns: true or false

Operations

In this section we show the Francy Callback Operations.

Callback

Creates a Callback object that holds a function and args to be executed by a trigger based on a trigger type. Please note, the Function must Return!

Callback(IsTriggerType, IsFunction, IsList(object)) - Returns: Callback

Examples:

```
gap> MyFunction := function() return "Hello World!"; end;
gap> callback := Callback(MyFunction);
gap> Id(callback);
```

Listing A.3: Francy simple Callback with no arguments

```
gap> MyFn := function(s) return Concatenation("Hello ", s); end;
gap> callback := Callback(MyFn);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> Add(callback, arg);
```

Listing A.4: Francy callback with one required argument

```
gap> MyFn := function(args) return args; end;
gap> callback := Callback(MyFn, ["Hello World"]);
```

Listing A.5: Francy callback with one known argument

```
gap> MyFn := function(a,b) return Concatenation(a, b); end;
gap> callback := Callback(MyFn, ["Hello "]);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> Add(callback, arg);
```

Listing A.6: Francy callback with default trigger type

```
gap> MyFn := function(a,b) return Concatenation(a, b); end;
gap> callback := Callback(TriggerType.CLICK, MyFn, ["Hello "]);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> Add(callback, arg);
```

Listing A.7: Francy callback with custom trigger type

```
gap> callback := NoopCallback();
```

Listing A.8: Francy no operation callback

```
gap> MyFn := function(a,b) return Concatenation(a, b); end;
gap> callback := Callback(TriggerType.CLICK, MyFn, ["Hello "]);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> SetTitle(arg, "Enter your name");
gap> Title(arg);
gap> Add(callback, arg);
gap> SetValue(arg, "Manuel"); # simulate the user input
gap> Value(arg);
gap> Trigger(GapToJsonString(Sanitize(callback)));
```

Listing A.9: Francy callback trigger function

NoopCallback

Creates an empty Callback object that does nothing. Useful to create menu holders.

NoopCallback() - Returns: Callback

RequiredArg

Creates a Callback RequiredArg. RequiredArg is user input driven and required for the execution of a Callback This value will be provided by the user.

RequiredArg(IsArgType, IsString(title)) - Returns: RequiredArg

Trigger

Triggers a callback function in GAP. Gets a JSON String object representation of the callback to execute.

Trigger(IsString(JSON)) - Returns: the result of the execution of the Callback defined Function

Add

Adds a RequiredArg to a specific Callback.

Add(IsCallback[, IsRequiredArg, List(IsRequiredArg)]) - Returns: Callback

Remove

Removes a RequiredArg from a specific callback.

Remove(IsCallback[, IsRequiredArg, List(IsRequiredArg)]) - Returns: Callback

Attributes

In this section we show the Francy Callback Attributes

Title

A title on a required arg is used to ask the user what is expected from his input.

Title(arg) - Returns: IsString with the title of the object

SetTitle

Sets the title of the required arg.

SetTitle(IsRequiredArg, IsString)

Value

Value(arg) - Returns: IsString with the value of the object

A value on a required arg is the actual input to be passed to gap. These values are stored as String for convenience, even if the ArgType specified for the RequiredArg is another. Explicit conversion is required within the Callbackfunction.

SetValue

Sets the value of the required arg.

SetValue(IsRequiredArg, IsString)

ConfirmMessage

A message to prompt to the user before triggering the callback.

ConfirmMessage(arg) - Returns: IsString with the value of the object

SetConfirmMessage

Sets the value of the confirmation message.

SetConfirmMessage(IsRequiredArg, IsString) - Sets the confirmation message

Francy Canvas

A Canvas is an area where the graphics representation of Francy live.

Please see Francy-JS for client implementation.

Categories

In this section we show the Francy Canvas Categories.

IsCanvas

Identifies Canvas objects.

IsCanvas(arg) - Returns: true or false

IsCanvasDefaults

Identifies CanvasDefaults objects.

IsCanvasDefaults(arg) - Returns: true or false

Representations

In this section we show the Francy Canvas Representations.

IsCanvasRep

Checks whether an Object has a Canvas internal representation.

IsCanvasRep(arg) - Returns: true or false

IsCanvasDefaultsRep

Checks whether an Object has a CanvasDefaults internal representation.

IsCanvasDefaultsRep(arg) - Returns: true or false

Operations

In this section we show the Francy Canvas Operations.

Canvas

Canvas(IsString(title)[, IsCanvasDefaults]) - Returns: Callback

Canvas represents a base element to draw graphics on. Inspired by the HTML canvas tag element which is used to draw graphics, in runtime, via JavaScript. Examples:

Create a simple Canvas:

```
gap> canvas := Canvas("");
gap> Id(canvas);
gap> SetTitle(canvas, "Quaternion Group Subgroup Lattice");
gap> Title(canvas);
gap> SetHeight(canvas, 400); # default 600
gap> Height(canvas);
gap> SetWidth(canvas, 400); # default 800
gap> Width(canvas);
gap> SetZoomToFit(canvas, false); # default true
gap> ZoomToFit(canvas);
gap> Draw(canvas);
```

Listing A.10: Francy simple canvas

Add

Adds a FrancyGraph to a specific Canvas. Well, the api is abstract enough to allow Adding a list of IsFrancyGraph to a canvas, but this results in setting the graph property only to the last IsFrancyGraph in the list.

Add(IsCanvas[, IsFrancyGraph, List(IsFrancyGraph)]) - Returns: Canvas

Remove

Removes a FrancyGraph from a Canvas.

Remove(IsCanvas[, IsFrancyGraph, List(IsFrancyGraph)]) - Returns: Canvas

Add

Adds a Chart to a specific Canvas. Well, the API is abstract enough to allow Adding a list of IsChart to a canvas, but this results in setting the graph property only to the last IsChart in the list.

Add(IsCanvas[, IsChart, List(IsChart)]) - Returns: Canvas

Remove

Removes a Chart from a Canvas.

Remove(IsCanvas[, IsChart, List(IsChart)]) - Returns: Canvas

Add

Adds a Menu to a specific Canvas.

Add(IsCanvas[, IsMenu, List(IsMenu)]) - Returns: Canvas

Remove

Removes a Menu from a Canvas.

Remove(IsCanvas[, IsMenu, List(IsMenu)]) - Returns: Canvas

Add

Adds a FrancyMessage to a specific IsCanvas.

Add(IsCanvas[, IsFrancyMessage, List(IsFrancyMessage)]) - Returns: IsCanvas

Remove

Removes a FrancyMessage from a specific IsCanvas.

Remove(IsCanvas[, IsFrancyMessage, List(IsFrancyMessage)]) - Returns: IsCanvas

Draw

Generates the JSON representation of the canvas and children objects

Draw(IsCanvas) - Returns: rec with json representation of the canvas

DrawSplash

Generates an HTML page and opens it within the default browser of the system

DrawSplash(IsCanvas) - Returns: rec with html generated

Attributes

In this section we show the Francy Attributes

Width

The Width of the canvas in pixels

Width(arg) - Returns: IsPosInt

SetWidth

Sets the Width of the canvas in pixels

SetWidth(IsCanvas, IsPosInt)

Height

The Height of the canvas in pixels

Height(arg) - Returns: IsPosInt

SetHeight

Sets the Height of the canvas in pixels

SetHeight(IsCanvas, IsPosInt)

ZoomToFit

ZoomToFit is a property that sets the zoom to fit behavior on change in the client implementation.

ZoomToFit(arg) - Returns: IsBool True if enabled otherwise False

SetZoomToFit

ZoomToFit is a property that sets the zoom to fit behavior on change in the client.

SetZoomToFit(IsCanvas, IsBool)

Title

A title on a required arg is used to ask the user what is expected from his input.

Title(arg) - Returns: IsString with the title of the object

SetTitle

Sets the title of the required arg.

SetTitle(IsCanvas, IsString)

TexTypesetting

Enables Tex typesetting on the client implementation

TexTypesetting(arg) - Returns: IsBool with the title of the object

SetTexTypesetting

Sets Tex typesetting on the canvas objects

SetTexTypesetting(IsCanvas, IsBool)

Francy Charts

It is possible to build Charts with simple Datasets. Currently, Francy, supports Bar, Line and Scatter Charts.

Please see Francy-JS for client implementation.

Categories

In this section we show the Francy Chart Categories.

IsChart

Identifies Chart objects.

IsChart(arg) - Returns: true or false

IsChartType

Identifies ChartType objects.

IsChartType(arg) - Returns: true or false

IsChartDefaults

Identifies ChartDefaults objects.

IsChartDefaults(arg) - Returns: true or false

IsAxisScaleType

Identifies AxisScaleType objects.

IsAxisScaleType(arg) - Returns: true or false

IsXAxis

Identifies XAxis objects.

IsXAxis(arg) - Returns: true or false

IsYAxis

Identifies YAxis objects.

IsYAxis(arg) - Returns: true or false

IsDataset

Identifies Dataset objects.

IsDataset(arg) - Returns: true or false

Representations

In this section we show the Francy Chart Representations.

IsChartRep

Checks whether an Object has a Chart internal representation.

IsChartRep(arg) - Returns: true or false

IsChartDefaultsRep

Checks whether an Object has a ChartDefaults internal representation.

IsChartDefaultsRep(arg) - Returns: true or false

IsChartTypeRep

Checks whether an Object has a ChartType internal representation.

IsChartTypeRep(arg) - Returns: true or false

IsAxisScaleTypeRep

Checks whether an Object has a AxisScaleType internal representation.

IsAxisScaleTypeRep(arg) - Returns: true or false

IsAxisRep

Checks whether an Object has a AxisRep internal representation.

IsAxisRep(arg) - Returns: true or false

IsDatasetRep

Checks whether an Object has a DatasetRep internal representation.

IsDatasetRep(arg) - Returns: true or false

Operations

In this section we show the Francy Chart Operations.

Chart

Chart(IsChartType[, IsChartDefaults]) - Returns: Chart

Every object to draw will be a subclass of this object. This will allow all the objects to contain the same base information.

Examples:

Create a simple Chart of type ChartType.LINE:

```
gap> chart:=Chart(ChartType.LINE);
gap> SetAxisXTitle(chart, "X Axis");
gap> SetAxisYTitle(chart, "Y Axis");
gap> data1 := Dataset("data1", [100,20,30,47,90]);
gap> Add(chart, data1);
```

Listing A.11: Francy simple line chart

Create a simple Chart of type ChartType.SCATTER:

```
gap> chart:=Chart(ChartType.SCATTER);
gap> SetAxisXTitle(chart, "X Axis");
gap> SetAxisYTitle(chart, "Y Axis");
gap> data1 := Dataset("data1", [100,20,30,47,90]);
gap> Add(chart, data1);
```

Listing A.12: Francy simple scatter chart

Create a simple Chart of type ChartType.BAR:

```
gap> chart:=Chart(ChartType.BAR);
gap> SetAxisXTitle(chart, "X Axis");
gap> AxisXTitle(chart);
gap> SetAxisXDomain(chart, ["domain1", "domain2", "domain3",
  ↪ "domain4", "domain5"]); # default []
gap> AxisXDomain(chart);
gap> SetAxisYTitle(chart, "Y Axis");
gap> AxisYTitle(chart);
gap> data1 := Dataset("data1", [100,20,30,47,90]);
gap> data2 := Dataset("data2", [51,60,72,38,97]);
gap> data3 := Dataset("data3", [50,60,70,80,90]);
gap> Add(chart, [data1, data2, data3]);
gap> Remove(chart, data1);
gap> Add(chart, data1);
gap> Remove(chart, [data2, data3]);
gap> Length(RecNames(chart!.data)) = 1;
```

Listing A.13: Francy simple bar chart

Add

Adds a Dataset to a specific Chart.

Add(IsChart[, IsDataset, List(IsDataset))) - Returns: Chart

Remove

Removes a Dataset from a specific Chart.

APPENDIX A. FRANCY USER MANUAL

Remove(IsChart[, IsDataset, List(IsDataset)]) - Returns: Chart

Dataset

Creates a dataset.

Dataset(IsString(title), IsList(data)) - Returns: Dataset

DefaultAxis

Returns the default settings for a ChartType

DefaultAxis(IsChartType) - Returns: rec

XAxis

Creates a XAxis

XAxis(IsAxisScaleType, IsString(title), IsList(domain)) - Returns: XAxis

YAxis

Creates a YAxis

YAxis(IsAxisScaleType, IsString(title), IsList(domain)) - Returns: YAxis

Attributes

In this section we show the Francy Attributes

ShowLegend

ShowLegend is a property that enables or disables the legend in the client implementation.

ShowLegend(arg) - Returns: IsBool True if enabled otherwise False

SetShowLegend

ShowLegend is a property that enables or disables the legend in the client implementation.

SetShowLegend(IsChart, IsBool)

AxisXTitle

This title is used to display the X Axis Title in the client implementation.

AxisXTitle(arg) - Returns: IsString with the title of the object

SetAxisXTitle

This title is used to display the X Axis Title in the client implementation.

`SetAxisXTitle(IsChart, IsString)`

AxisYTitle

This title is used to display the Y Axis Title in the client implementation.

`AxisYTitle(arg)` - Returns: `IsString` with the title of the object

SetAxisYTitle

This title is used to display the Y Axis Title in the client implementation.

`SetAxisYTitle(IsChart, IsString)`

AxisXDomain

This is the domain of the X Axis values in the client implementation.

`AxisXDomain(arg)` - Returns: `IsList`

SetAxisXDomain

This is the domain of the X Axis values in the client implementation.

`SetAxisXDomain(IsList, IsList)`

Francy Core

Francy is responsible for generating JSON metadata which allows external tools / libraries / frameworks to add a visual representation. This JSON representation defines the contract between this package (server) and a GUI framework (client), this enables complete SoC (Separation of Concerns). Francy can be used to provide a graphical interactive environment on existing GAP packages.

A JSON schema is present and can be used to produce clients for this package. See `schema/francy.json`

To map required / optional attributes from the schema into GAP code, the implementation follows the following criteria:

Object creation requests mandatory attributes, i.e. required with no default value, e.g. `canvas:=Canvas("Title")`

Object creation accepts an object of defaults, i.e. default values for mandatory fields but that might repeat throughout the creation of multiple similar objects, e.g. `defaults:=DefaultCanvas; defaults!.zoomToFit:=false; canvas:=Canvas("Title",defaults);` Where `DefaultCanvas` contains defaults for width (800) and height (600)

APPENDIX A. FRANCY USER MANUAL

Attributes associated with the object that can be set, .i.e. optional with no defaults, e.g. `canvas:=Canvas("Title"); SetTexTypesetting(canvas,true);`

The API follows a common convention and adding objects to other objects is done using `Add(objectHolder,object)`

Although Francy has the concept of a Graph, it does not implement any Mathematics Graphs Theory.

Please see Francy-JS for client implementation.

Categories

In this section we show the Francy Core Categories.

IsFrancyObject

Identifies all Objects in Francy.

`IsFrancyObject(arg)` - Returns: true or false

IsFrancyDefaultObject

Identifies all Default records in Francy.

`IsFrancyDefaultObject(arg)` - Returns: true or false

IsFrancyTypeObject

Identifies all Type records in Francy.

`IsFrancyTypeObject(arg)` - Returns: true or false

Attributes

In this section we show the Francy Core Attributes

FrancyId

All Objects created in Francy have a generated identifier.

`FrancyId(arg)` - Returns: `IsString` with the id of the object

FrancyId

Prints the object unique identifier.

`FrancyId(arg1)` - Returns: `IsString` with the id of the object

SetFrancyId

Use with care! Changing the unique ID might be useful in certain cases, but bare in mind it might cause unexpected behavior if you're not sure about!

`SetFrancyId(o, s)`

Francy Graphs

It is possible to build Graphs, directed or undirected.

Please see Francy-JS for client implementation.

Categories

In this section we show the Francy Graph Categories.

IsFrancyGraph

Identifies Graph objects.

IsFrancyGraph(arg) - Returns: true or false

IsFrancyGraphType

Identifies GraphType objects.

IsFrancyGraphType(arg) - Returns: true or false

IsFrancyGraphDefaults

Identifies GraphDefaults objects.

IsFrancyGraphDefaults(arg) - Returns: true or false

IsShape

Identifies Shape objects.

IsShape(arg) - Returns: true or false

IsShapeType

Identifies ShapeType objects.

IsShapeType(arg) - Returns: true or false

IsShapeDefaults

Identifies ShapeDefaults objects.

IsShapeDefaults(arg) - Returns: true or false

IsLink

Identifies Link objects.

APPENDIX A. FRANCY USER MANUAL

IsLink(arg) - Returns: true or false

IsLinkDefaults

Identifies LinkDefaults objects.

IsLinkDefaults(arg) - Returns: true or false

Representations

In this section we show the Francy Graph Representations.

IsFrancyGraphRep

Checks whether an Object has a Graph internal representation.

IsFrancyGraphRep(arg) - Returns: true or false

IsFrancyGraphDefaultsRep

Checks whether an Object has a GraphDefaults internal representation.

IsFrancyGraphDefaultsRep(arg) - Returns: true or false

IsFrancyGraphTypeRep

Checks whether an Object has a GraphType internal representation.

IsFrancyGraphTypeRep(arg) - Returns: true or false

IsShapeRep

Checks whether an Object has a Shape internal representation.

IsShapeRep(arg) - Returns: true or false

IsShapeDefaultsRep

Checks whether an Object has a ShapeDeafults internal representation.

IsShapeDefaultsRep(arg) - Returns: true or false

IsShapeTypeRep

Checks whether an Object has a ShapeType internal representation.

IsShapeTypeRep(arg) - Returns: true or false

IsLinkRep

Checks whether an Object has a Link internal representation.

IsLinkRep(arg) - Returns: true or false

IsLinkDefaultsRep

Checks whether an Object has a LinkDeafults internal representation.

IsLinkDefaultsRep(arg) - Returns: true or false

Operations

In this section we show the Francy Graph Operations.

Graph

Graph(IsFrancyGraphType[, IsFrancyGraphDefaults]) - Returns: Graph

Every object to draw will be a subclass of this object. This will allow all the objects to contain the same base information.

Examples:

Create a simple Graph of type GraphType.DIRECTED and simple Shapes connected with Links:

```
gap> graph := Graph(GraphType.DIRECTED);
gap> SetSimulation(graph, false);
gap> shape1 := Shape(ShapeType.SQUARE);
gap> shape1!.layer := 1;
gap> shape2 := Shape(ShapeType.TRIANGLE);
gap> shape2!.layer := 3;
gap> link := Link(shape1, shape2);
gap> Add(graph, link);
gap> Add(graph, [shape1, shape2]);
```

Listing A.14: Francy simple directed graph

Create a simple Graph of type GraphType.UNDIRECTED and a simple Shape with a TriggerEvent.RIGHT_CLICK Callback:

UnsetNodes

UnsetNodes(arg) - Removes all nodes from gaph

UnsetLinks

UnsetLinks(arg) - Removes all nodes from gaph

Add

Add IsLink to a specific Graph.

APPENDIX A. FRANCY USER MANUAL

```
gap> graph := Graph(GraphType.UNDIRECTED);
gap> shape := Shape(ShapeType.SQUARE);
gap> MyFn := function()
  Add(graph, Shape(ShapeType.Circle)); return graph;
end;
gap> callback := Callback(TriggerType.RIGHT_CLICK, MyFn);
gap> Add(shape, callback);
gap> Add(graph, shape);
```

Listing A.15: Francy simple undirected graph

Add(IsFrancyGraph[, IsLink, List(IsLink)]) - Returns: Graph

Remove

Remove IsLink from a specific Graph.

Remove(IsFrancyGraph[, IsLink, List(IsLink)]) - Returns: Graph

Add

Add IsShape to a specific Graph.

Add(IsFrancyGraph[, IsShape, List(IsShape)]) - Returns: Graph

Remove

Remove IsShape from a specific Graph.

Remove(IsFrancyGraph[, IsShape, List(IsShape)]) - Returns: Graph

Shape

Every object to draw will be a subclass of this object. This will allow all the objects to contain the same base information.

Shape(IsShapeType[, IsString(title), IsShapeDefaults]) - Returns: Shape

GetShape

Gets a Shape node from a graph by ID.

GetShape(IsFrancyGraph, IsString) - Returns: Shape

GetShapes

Gets a Shape node from a graph by ID.

GetShapes(IsFrancyGraph, IsString) - Returns: List(Shape)

Add

Add Menu to a specific Shape.

Add(IsShape[, IsMenu, List(IsMenu)]) - Returns: Shape

Remove

Remove Menu from a specific Shape.

Remove(IsShape[, IsMenu, List(IsMenu)]) - Returns: Shape

Add

Add Callback to a specific Shape.

Add(IsShape[, IsCallback, List(IsCallback)]) - Returns: Shape

Remove

Remove Callback from a specific Shape.

Remove(IsShape[, IsCallback, List(IsCallback)]) - Returns: Shape

Add

Add Callback to a specific Shape.

Add(IsShape[, IsFrancyMessage, List(IsFrancyMessage)]) - Returns: Shape

Remove

Remove Callback from a specific Shape.

Remove(IsShape[, IsFrancyMessage, List(IsFrancyMessage)]) - Returns: Shape

Link

Creates a Link between the two Shapes.

Link(IsShape, IsShape) - Returns: Link

Links

Creates a Link between the Shape of the first list and the second list.

Links(List(IsShape), List(IsShape)) - Returns: List(Link)

GetLink

Gets a Link from a graph by ID.

APPENDIX A. FRANCY USER MANUAL

GetLink(IsFrancyGraph, IsString) - Returns: Link

GetLinks

Gets a Link from a graph.

GetLinks(IsFrancyGraph, IsString) - Returns: List(Link)

Attributes

In this section we show the Francy Core Attributes

Title

Sets the title on the Shape. Supports LaTeX syntax that gets translated, if enabled on the client.

Title(arg) - Returns: IsString with the title of the object

SetTitle

Sets the title of the Shape.

SetTitle(IsRequiredArg, IsString)

Color

The Color of the current shape.

Color(arg) - Returns: IsInt

SetColor

Sets the Color value.

SetColor(IsShape, IsString)

PosX

The Position in the X Axis of the Shape in the Canvas in pixels

PosX(arg) - Returns: IsInt

SetPosX

Sets the Position in the X Axis of the Shape in the Canvas in pixels

SetPosX(IsShape, IsInt)

PosY

The Position in the Y Axis of the Shape in the Canvas in pixels

PosY(arg) - Returns: IsInt

SetPosY

Sets the Position in the Y Axis of the Shape in the Canvas in pixels

SetPosY(IsShape, IsInt)

Size

The Size of the Shape

Size(arg) - Returns: IsPosInt

SetSize

Sets the Size of the Shape

SetSize(IsShape, IsPosInt)

Layer

The Layer in which the node will be placed. This property is also used to apply a color based on a scale

Layer(arg) - Returns: IsInt

SetLayer

Sets the Layer number.

SetLayer(IsShape, IsInt)

ParentShape

The ParentShape in which the node will be placed. This property is also used to apply a color based on a scale

ParentShape(arg) - Returns: IsShape

SetParentShape

Sets the ParentShape.

SetParentShape(IsShape, IsShape)

Simulation

APPENDIX A. FRANCY USER MANUAL

Simulation is a property that sets the simulation behavior by applying forces to organize the graphics, without the need to provide custom positions, in the client implementation.

Simulation(arg) - Returns: IsBool True if enabled otherwise False

SetSimulation

Sets the Simulation behavior.

SetSimulation(IsCanvas, IsBool)

Collapsed

Collapsed is a property that sets to collapsed the graphic structure by default

Collapsed(arg) - Returns: IsBool True if enabled otherwise False

SetCollapsed

Sets the Collapsed behavior.

SetCollapsed(IsCanvas, IsBool)

Selected

Collapsed is a property that sets to collapsed the graphic structure by default

Selected(arg) - Returns: IsBool True if enabled otherwise False

SetSelected

Sets the Collapsed behavior.

SetSelected(IsCanvas, IsBool)

ConjugateId

Collapsed is a property that sets to collapsed the graphic structure by default

ConjugateId(arg) - Returns: IsBool True if enabled otherwise False

SetConjugateId

Sets the Collapsed behavior.

SetConjugateId(IsCanvas, IsBool)

Weight

The Weight of the current link.

Weight(arg) - Returns: IsInt

SetWeight

Sets the Weight value.

SetWeight(IsLink, IsInt)

Length

The Length of the current link.

Length(arg) - Returns: IsInt

SetLength

Sets the Length value.

SetLength(IsLink, IsInt)

Invisible

The Invisible of the current link.

Invisible(arg) - Returns: IsBoolean

SetInvisible

Sets the Invisible value.

SetInvisible(IsLink, IsBool)

Color

The Color of the current link.

Color(arg) - Returns: IsInt

SetColor

Sets the Color value.

SetColor(IsShape, IsString)

Title

The Title of the current link.

Title(arg) - Returns: IsInt

SetTitle

Sets the Title value.

SetTitle(IsShape, IsString)

Francy Menus

Menus are agregators of actions that are represented here by Callbacks. Menus can have SubMenus, and are constituted by a Title and a Callback.

Please see Francy-JS for client implementation.

Categories

In this section we show the Francy Menu Categories.

IsMenu

Identifies Menu objects.

IsMenu(arg) - Returns: true or false

Representations

In this section we show the Francy Menu Representations.

IsMenuRep

Checks whether an Object has a Menu internal representation.

IsMenuRep(arg) - Returns: true or false

Operations

In this section we show the Francy Menu Operations.

Menu

Creates a Menu for a Callback Is up to the client implementation to sort out the Menu and invoke the Callback

Menu(IsString(title)[, IsCallback]) - Returns: Menu

Add

Add Menu to a specific Menu creating a Submenu. Is up to the client implementation to handle this.

Add(IsMenu[, IsMenu, List(IsMenu)]) - Returns: Menu

Remove

Remove Menu from a specific Menu. The client should be able to handle this.

Remove(IsMenu[, IsMenu, List(IsMenu)]) - Returns: Menu

Attributes

In this section we show the Francy Core Attributes

Title

A title on a Menu is used to identify the menu entry.

Title(arg) - Returns: IsString with the title of the object

SetTitle

Sets the title of the Menu.

SetTitle(IsMenu, IsString)

Francy Messages

FrancyMessage is an object that holds a message.

These messages can be used to provide information to users in the form of SUCCESS, INFO, WARNING, ERROR. Please see Francy-JS for client implementation.

Categories

In this section we show the Francy FrancyMessage Categories.

IsFrancyMessage

Identifies FrancyMessage objects.

IsFrancyMessage(arg) - Returns: true or false

IsFrancyMessageType

Identifies MessageType objects.

IsFrancyMessageType(arg) - Returns: true or false

Representations

In this section we show the Francy FrancyMessage Representations.

IsFrancyMessageRep

Checks whether an Object has a FrancyMessage internal representation.

IsFrancyMessageRep(arg) - Returns: true or false

IsFrancyMessageTypeRep

Checks whether an Object has a FrancyMessage internal representation.

IsFrancyMessageTypeRep(arg) - Returns: true or false

Operations

In this section we show the Francy FrancyMessage Operations.

FrancyMessage

Adds an info label with the format label: value

FrancyMessage(IsString, IsString) - Returns: FrancyMessage

Attributes

In this section we show the Francy Core Attributes

Title

A title on a FrancyMessage is used to display the title information to the user.

Title(arg) - Returns: IsString with the title of the object

SetTitle

Sets the title of the FrancyMessage.

SetTitle(IsFrancyMessage, IsString)

Value

A value on a FrancyMessage is used to display the information to the user.

Value(arg) - Returns: IsString with the title of the object

SetValue

Sets the actual message of the FrancyMessage.

SetValue(IsFrancyMessage, IsString)

Francy Util

Operations

In this section we show the Francy Util Operations. Contains utility methods to handle Object printing/viewing, Sanitizing, etc.

JUPYTER_ViewString

This method will pretty print in jupyter environment.

JUPYTER_ViewString(arg) - Returns: String

Sanitize

This method will clone a Object and return a sanitized record, traversing all the components and sanitizing when appropriate. Sanitizing in this context means, replace everything with it's string representation that can't be converted into JSON!

Sanitize(IsObject) - Returns: rec

MergeObjects

This method will merge the properties of 2 IsFrancyObjects into one rec.

MergeObjects(IsFrancyObject, IsFrancyObject) - Returns: rec

GenerateID

This method will generate a sequential ID for use as object identifier.

GenerateID() - Returns: IsString