Diabolic Magic Squares

Marcela Ortega Soto, 2014053916
Fabián Solano Rodriguez, 2013116291

September 4, 2015

1 Index

2 Introduction

This paper consists of a detailed description about the investigation and development of a project called Diabolic Magic Squares. The project has to be developed in GNU/Linux, using Prolog as a programming language for the logic part and a different programming language of a different paradigm for the Grapichal User's Interface.

One principal objective is to manage the logical paradigm by doing a project that make us think in how to solve problems using this paradigm.

For this project the programming language chosen was Prolog. This is the most popular logical language. As a complement, the project integrates the binding or joint from two different language, in this case Prolog and Java. Java was chosen to build the grapichal interface.

3 Problem description

The project consists of developing a two way program that validates, and generates 4x4 diabolic magic squares. The program will contain a backend performed in prolog which will handle the logic of the validation and generation, and it will also have a frontend which will handle the I/O interaction with the user.

A "diabolic magic square", also known as "pandiagonal magic square" or "panmagic square", is a "n x n" matrix whose cells contain the numbers 1,2,..,(n2), arranged in such a way that the cells in every row, column, and diagonal sum to (n3 + n)/2.

Diagonals are broadly defined: they include not only the major (corner-to-corner) diagonals, but the "broken" diagonals as well. If you imagine tiling a wall with a large number of identical copies of a diabolic magic square, every set of four cells in a line, horizontally, vertically, or diagonally, will add up to the same number.

$4x4 Diabolic Magic Square : The magic constant 34((n^3 + n)/2)$can be seen in a number of patterns in addition to the rows, columns and diagonals.

- Any of the sixteen 2x2 squares, including those that wrap around the edges of the whole square.
- The corners of any $3x3$ square.
- Any pair of horizontally or vertically adjacent numbers, together with the corresponding pair displaced by a (2, 2) vector.

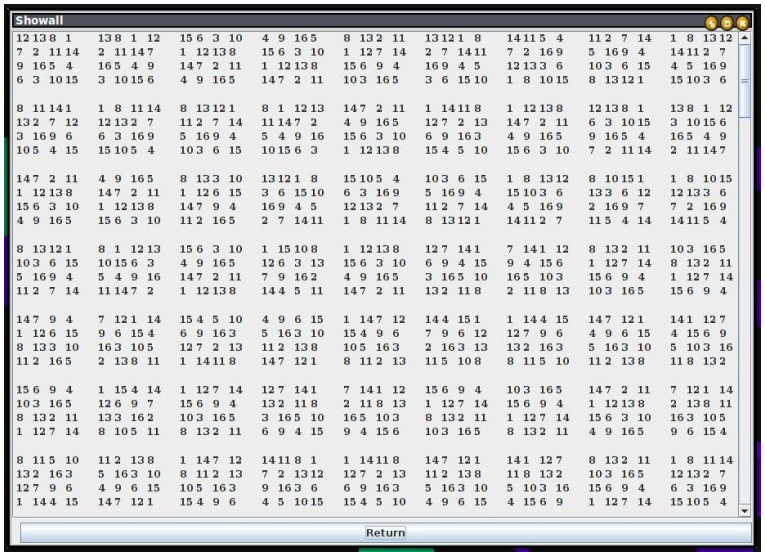
4 User's guide

Requirements: A computer using Linux as main OS, with free memory. This computer must have Swi-Prolog and JPL Library installed. It is recommended to have an 64 bits processor for a better performance of the program. Besides this programs, it is necessary to install Intellij IDE distribution for Java. Steps:

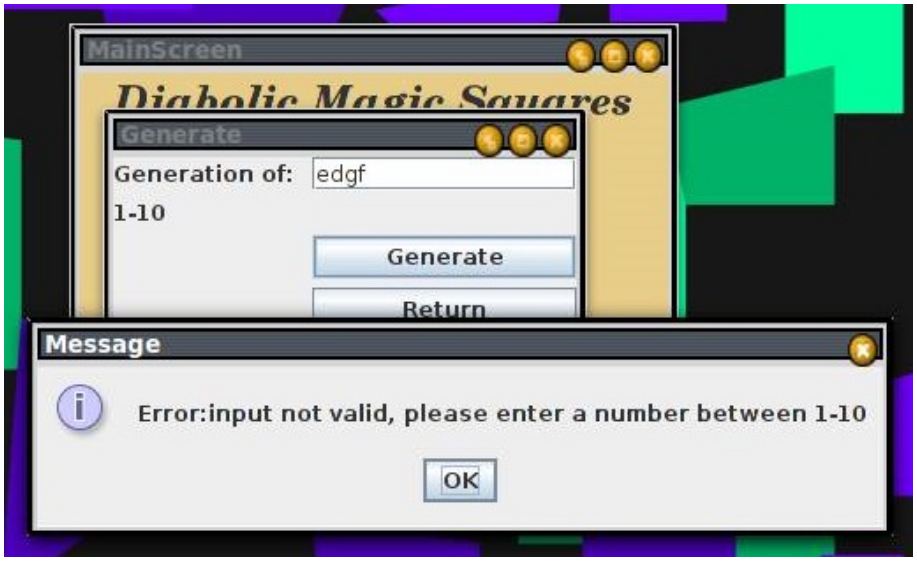
1. Initiate the computer
2. Initiate Eclipse with Java and JPL
3. Open the project
4. Click on the "Run" button
5. Select what you want to do
6. Close

- The past image shows the main screen of the user interface of our project.

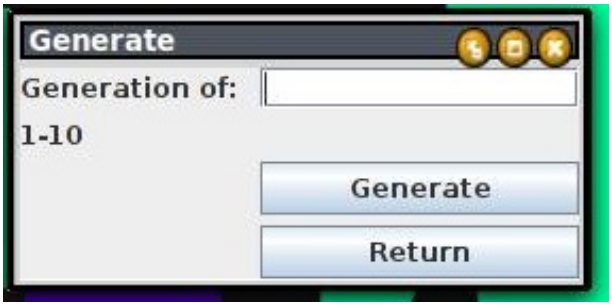
- The next picture shows how the showall works.



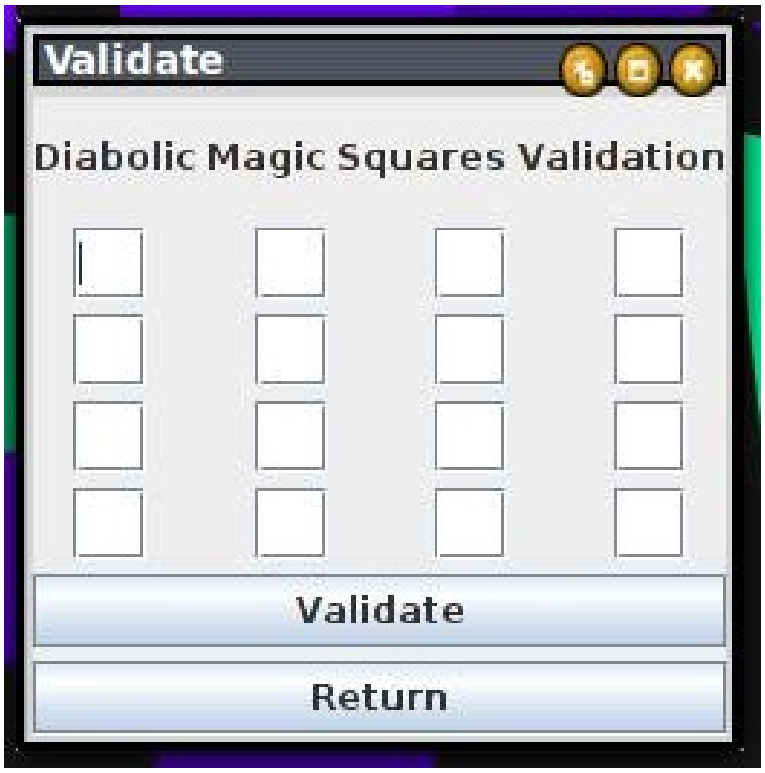
- The next picture shows how the error are notified to the user.



- The next image shows how generate works.



- The next image shows how validate works.



5 Development environment

The project was developed in Linux, Ubuntu.

Linux is a Free Software Operative System based on GNU.

We also used Swi-Prolog as our IDE and interpreter.

SWI-Prolog offers a comprehensive free Prolog environment. Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications. SWI-Prolog is widely used in research and education as well as commercial applications.

For the connection of Java and Prolog we used a library called JPL. This library provides all the utilities for managing code in Prolog but also in Java.

We choose this library because is the one better documented and easy to use. It has many functional examples with connections between this two languages. We thought this will be the most useful library for developing this project.

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, nowadays almost everywhere.

The IDE chosen to programm in Java was Intellij. IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software. It is developed by JetBrains (formerly known as IntelliJ).

- System Requirements:

1. Windows: 10/8/7/Vista/2003/XP — MacOS X: 10.5 - 10.9 — Linux: GNOME or KDE desktop
2. 1 GB RAM minimum, 2 GB RAM recommended
3. 300 MB hard disk space + at least 1 GB for caches
4. 1024x768 minimum screen resolution
5. JDK 1.6 or higher

This IDE was chosen for its many tools which help and aid the programmer work. Another great reason for choosing this IDE, was the simple and esay way to do a grapichal user interface. Just by dragging and placing different components in place a total and functional GUI could be achieved.

6 Program Design

Design details We decided to use the basic Prolog functions and module the program in the most simple way.

We used a design technique that consists in using four different predetermine squares and combine them changing each value for a 1, 2, 4 or 8.

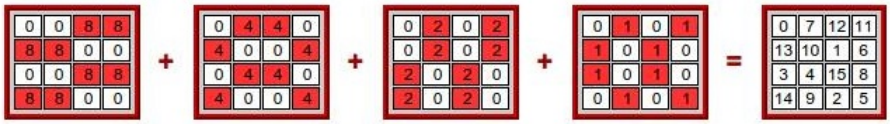
This method aloud us to create the first 24 squares. Using this squares we could create and complete the 384 final squares.

At the end, using the five different methods: reflection, convolution, rotation to the center, rotation rows and rotation columns; we could generate the entire list of diabolic magic squares.

Algorithms For validating if a square is diabolic, we used the algorithm to know what the adding of the rows, the columns and the diagonal should be. This algorithm result for 4x4 Diabolic Magic Squares is 34.

All of the Order 4 Pan-Magic Squares are based on the same underlying "Magic Carpet", a simple pattern consisting of alternating pairs of ones with zeros.

Four samples of this pattern are multiplied by 8, 4, 2, and 1, to make the Magic Carpets which are added together to make the final square. The 8, 4, 2, and 1, can be used in any order to make different squares.



Functions and Data Structures in the Frontend As the frontend was done in Java it is made of different classes and each class has their own methods. The main class contains the call to the methods:

- Frontend.connection(); This method starts the connection with Prolog.
- MainScreen.start(); This method starts the GUI.

The class frontend contains the methods:

- connection(); This method Enables the connection to Prolog.
- verifySquare(); This method verifies the input actually is a diabolic magic square.
- showall(); This method show all the possible diabolic magic sqaures.
- generate(); This method Generates the amount of squares requested.

The class mainscreen controls the calls to the other screens regarging the user's decision. The classes of validate, generation and showall contains the methods proper to the grapichal interface and some general verifications that the user only inputs numbers and not letters or leave some blanket spaces.

Predicates, Data Structures and Rules in the Backend

- *select*: Divides the rows of the square.
- diabolic: It has two functions: the first one is validating if a square is diabolic, the second one generates the specific number of requested diabolic magic squares.
- add1: Adds one to the entire square.
- multi: Multiplies a number and a list.
- addList: Adds two lists.
- square: Generates a diabolic magic square.
- invert: Inverts a list.
- reflection: Applies reflection to a square.
- rotationCenter: Applies rotation to the center.
- rotationRow: Applies rotation of rows.
- rotationColumn: Applies rotation of columns.
- convolution: Applies convolution to a square.
- showall: Generates the 384 squares.
- *sum*: *Addsthevaluesofarowandvalidate*

Why did we choose Java? We choose Java for its suitability in this proyect and simplicity to design and build a graphical interface. One of the main reasons Java was the programming language chosen for the frontend, was the perfect connection it has with Prolog through the library jpl. This made the work smoother and made possible to fulfill all the requierements established.

User Interface Design The design choosen was a simple yet effictive interface. The idea proposed, which was executed, was to guide the user throughout all the experience with the program. To accomplish this, the interface consists of several screens with each of the functions and requirements. It has a different screen to verify, to generate and to show all. Also, every time the user enters a wrong input the programs displays friendly error messages indicating the error and indicating the user what the right input is.

7 Student activity log

During the creation of a project in group, the overriding should be the division and distribution of task in such a way that every member work equally to reach the goal set.

In the project Diabolic Magic Squares each of the two members worked arduously with the purpose to deliver the project.

Fabian Solano Rodriguez		
Actividad	**Descripcion**	**Horas invertidas**
Instalacion	Eclipse, SWI Prolog, JPL.	2
Investigación	Funcionamiento de Cuadros mágicos diabólicos.	6
Lenguaje Prolog	Estudio de sintaxis y semantica. Funcionamiento, variables, etc.	2
Generador de cuadros	Desarrollo del programa que genera cuadros mágicos diabólicos.	2
Desarrollo de algoritmos	Desarrollo de algoritmos a aplicar sobre los cuadros.	12
Documentación	Elaboración de la documentación del proyecto.	4
Desarrollo de soluciones	Planteamiento de las posibles soluciones a cada problema.	11
Conexión de lenguajes	Conexión entre Java y Prolog.	6
	Total	45

Marcela Ortega Soto		
Actividad	**Descripcion**	**Horas invertidas**
Instalacion	Eclipse, SWI Prolog, JPL.	2
Investigación	Funcionamiento de Cuadros mágicos diabólicos.	4
Lenguaje Prolog	Estudio de sintaxis y semantica. Funcionamiento, variables, etc.	6
Generador de cuadros	Desarrollo del programa que genera cuadros mágicos diabólicos.	10
Desarrollo de algoritmos	Desarrollo de algoritmos a aplicar sobre los cuadros.	8
Documentación	Elaboración de la documentación del proyecto.	3
Desarrollo de soluciones	Planteamiento de las posibles soluciones a cada problema.	6
Conexión de lenguajes	Conexión entre Java y Prolog.	6
	Total	45

8 Project final status

The project, after an arduous work, was completed with the basic functions, the user interface and the console commands in SWI-prolog. One of the hardest challenge was to fully understand the syntaxis Prolog uses and managing the progamming language due to the fact of the lack of a proper IDE for this language.

It completes with the following:

- Start
- showall
- square1
- reflection
- convolution
- rotationCenter
- rotationRow
- rotationColumn
- diabolic

The limitations is to specify the errors produced by the user and indicating how to fixed them. Another limitation is to fully verify that there are no repeated matrices while executing the showall.

9 Conclusions

1. A deep study is require in order to fully understand Prolog syntax.
2. Using Java with Prolog assures a perfect binding with two different programming paradigms.
3. Intellij eases the programmers load regarding the building of a GUI, due to the tools it provides. It allows to drag and place different components
4. Learning programming languages in a short time can be really difficult and hard to adapt.
5. Managing time is very important for a good developing of any project.
6. Investigation is one of the most important parts when you are working in a project.

10 Suggestions and recommendations

1. It is better to use an IDE easy to learn.
2. Dividing the time expend in every aspect of the project is important. This is how you know what you should do first.
3. Use Intellij to program in Java. It's simple, precise, effective but above all easy to use.
4. Use a Version Control System for avoiding the lost of code.
5. For a better project use always recursion. In that way you can pass variables between functions.

11 References

- 3 formas de resolver un cuadro mágico - wikiHow (s. f.). Recuperado el 04 de Septiembre del 2015, de http://es.wikihow.com/resolver-un-cuadro-m
- $Calvo, S. y Perez, F. (2001). Cuadrados_magicos_4x4 Recuperado el 04 de Septiembre del 2015, de http://recursostic.educacion.es/descartes/web/materiales_didacticos/CuadMag/$
- Grogono Magic Squares (2010). Recuperado el 04 de Septiembre del 2015, de http://www.grogono.com/magic/4x4.php
- Prolog Ejemplos - Documents (s. f.). Recuperado el 04 de Septiembre del 2015, de http://myslide.es/documents/prolog-ejemplos.html
- Github Inc (2015). Packages-jpl/examples/java/Family at master · SWI-Prolog/packages-jpl · GitHub Recuperado el 04 de Septiembre del 2015, de https://github.com/SWI-Prolog/packages-jpl/tree/master/examples/java/Family
- Sewell, D. (2003). Magic Squares Recuperado el 04 de Septiembre del 2015, de http://www.halexandria.org/dward090.htm
- Stack Exchange Inc (2015). Connect between java and swi prolog - Stack Overflow Recuperado el 04 de Septiembre del 2015, de http://stackoverflow.com/questions/