Second Project: ¡Name¿.

Santiago, Gamboa Ramírez
santigr17@gmail.com

Juan Esteban, Navarro Camacho
jnavcamacho@gmail.com

April 16, 2016

Abstract

This project will discuss the usage of the logical programming language called Prolog to solve a diabolic magic squares made with 4 by 4 squares and Python as a front-end. A diabolic magic squares are matrix of 4 rows and 4 columns which have unique numbers on each cell from 1 to 16. This project will implement the solution of this problem using logical deduction.

1 Introduction

A diabolic magic squares are an special matrix made by n rows and n columns and each cell of this matrix has an unique value from 1 to n^2. This matrix in some cases has an interesting behavior. This behavior makes to this matrix have some mathematical properties; form example the sum of each cell to make a row has the same result as the others.
The diagonals have the same sum like the rows, the sum of all values from a column have the same value as the rows or diagonal have.

Figure 1: Making queries into a table.

2 Objectives

3 Problem description

On the image 1 it shows a diagonal magic square, and if the user make a sum of the cells of some rows or some column the result will be the same for both expressions. There are more conditions that proves this square as magic or diabolic.
For example:

4 Requirements

For the implementation of this project, its necessary to create 2 programs that will be interacting with each other.

- A Frontend, which purpose will be to interact with the user , take care of I/O operations and communicate with the backend.

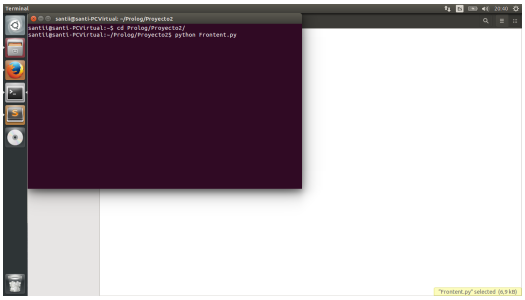

Figure 2: Command to run the program

Figure 3: This is the main window of the program, here the user can see the grid to insert a square, also has the option to ask for an amount of Diabolic Magic Squares

- A backend, where all the diabolic magic square resolution is going to happen.
- This backend will be receiving the entries for the execution of the predicates.
- The Frontend will be developed in any language chosen by the student, it has to connect with the backend (Prolog) to pass over to the Prolog program the operation requested by the user.

5 User Guide

The next series of images illustrate the use of the program. The following images show the different kind of errors that can occur.

Figure 4: This message show when the button "verify" is pressed.

Figure 5: This message show when the button "verify" is pressed.

Figure 6: This message show when the button "verify" is pressed.

Figure 7: This message show when the button "verify" is pressed.

Figure 8: This message show when the button "verify" is pressed.

Figure 9: This message show when the button "show" is pressed.

Figure 10: This message show when the button "verify" is pressed.

Figure 11: This is a warning for the user in order to ask for patience when the button "show" is pressed.

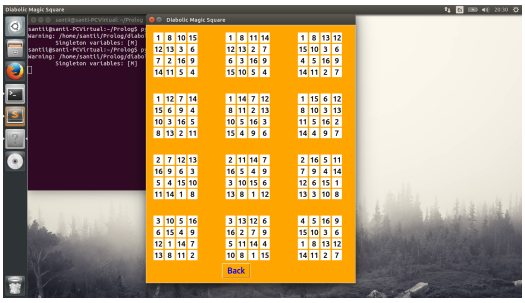

Figure 12: This window is displayed when the button "show" is pressed and the value is correct.

Figure 13: Here the program asks to the user if he wants to continue to a building part of the program.

Figure 14: SWI-Prolog environment on GNU-Linux

6 Development environment

6.1 SWI-Prolog

This framework offers and easy way to implement prolog queries in terminal, it is compatible with the most common programming languages and development environments like Java, Python and Netbeans as an IDE.

6.2 Python with PySwip

PySWIP requires SWI-Prolog as a shared library to make an interface to make queries from Python to Prolog.
To install PySwip this tutorial will show how to: `https://pyswip.googlecode.com/svn-history/r129/trunk/INSTALL`

6.3 JPL for Java

JPL is a set of Java classes and C functions providing an interface between Java and Prolog. This framework uses Java Native Interface to work with the prolog queries. This framework has to types of interfaces, the low level and the high level.

- Low Level: For C and C++ development.
- High Level: For Java programmers.

6.4 NetBeans Java

There is another option to make queries to from from another programming language, this language is called Java.For Netbeans IDEs, after install jpl libraries the user must to include the .jar files.

7 Program design

The program consists of two main parts, the first is the logical part responsible for calculating the Diabolic Magic Squares, it is composed of a function that checks whether a list is a Diabolic Magic Square and returns true or false. Part two consists fronted, or user interface, which query logic to the outcome of the question asked by the user. It also has the function of displaying pictures, which are displayed in a new window, depending on the amount desired by the user.

8 Data Structures and functions

8.1 User Interface

class guiDMsquares: with this class we manage the windows and create all the interface
function verify: this function valid all the entries and if these are right concatenates the values.
function suqares1-10: Here is where we construct the squares and place them in the new window, this function also calls the query to ask prolog for the

squares.
function error: We manage the errors by a code, in the message is the causes of the error and we unify the errors in one function.
function random: With this processor we fill all the grid with random values.
function clear: It is in charge to clean the values of the entries grid.

8.2 Diabolic.pl

function diabolic: This function is called when we need to verify a list.
function diabolico: This other function return all the possible Diabolic Magic Squares.

9 Student activity log

Date	Juan's Task Description	Spe
8/April/2016	Design and understanding the project	4
10/April/2016	Deciding which language is better for the project	3
11/April/2016	Understanding prolog using swi prolog	2
12/April/2016	Research diabolic magic squares, how to verify one	14
13/April/2016	Design the principal menu with Santiago and simple functions	10
14/April/2016	Research how to use Java jpl library	4
15/April/2016	Fixing errors with pyswip, figure it out how to communicate in other way.	4
	Total	54

Date	Santiago's Task Description
06/April/2016	Installing swi prolog and looking for othe
07/April/2016	Try some functions. Install pyswip and
09/April/2016	Create a simple interface
10/April/2016	Meeting with Juan to define the structures of the table. Scheme and register. St
11/April/2016	Create a method that gets user entries and pu
12/April/2016	Error connecting python and swi pro
14/April/2016	Trying to install java jpl for proje
15/April/2016	finish the interface and doing documen
	Total

10 Project final status

This program has successfully verify and query for less than 26 squares. The communication between python and prolog is slow and take a while. The interface has a good validation of the entries and give feedback or message errors to the user.

10.1 Bugs

If you consult for a number of squares and them try to verify another square the query fails.

10.2 Not resolved Problems

Show all, the method to show all the possible solutions for a Diabolic Magic Square stops when solves the 26th square.

10.2.1 Casting List from Prolog to Python

This is the main not resolved problem:
When the user types to get a Diabolic Magic Square using Prolog from terminal the result will be like this example. Getting a diabolic magic square :

```
$ prolog diabolic.pl
$ ?-diabolico(X).
$ X = [1, 8, 10, 15, 12, 13, 3, 6, 7|...] [write]
$ X = [1, 8, 10, 15, 12, 13, 3, 6, 7, 2, 16, 9, 14, 11, 5, 4]
```

But in python when the user wants to include this line into a query on Python, the result will be. Print on Python

```
$ python  test.py
>> {X = Variable (16), Variable(16),...}
>>
```

This means that prolog array of int can't be displayed using a normal casting.

10.2.2 Including .jar Import to Link Java and Prolog

After intalling swi-prolog on linux, netbeans can't link the jar to the file.

10.2.3 Pwsip install on Linux Mint

The make file to install the swi-prolog made some erros on Juan's computer, just Santiago's can run pyswip with python

11 Conclusions

There is no enough updated documentation to the usage of Python with Pyswip for complex Prolog and Python Applications. It means for complex projects Python is not a good option.

12 Suggestions and recommendations

For JPl and Java development. As the site web says: currently JPL only supports the embedding of a Prolog engine within the Java VM. For many further versions this API will be available for C and C++ development.

Python and PySwip are good options to implement queries if the code on prolog has the facts defined, but it the user wants to implement a query with a prolog returned variables is not warrantied the success of the app.

If the user wants to implement Netbeans and JPL: Ensure that the Linux distribution doesn't have OpenJDK intalled this is because JPL works with JDK private version, the non-free.

To install the correctly version of JDK the user needs to go the the following web site and follow every step described there. This web site has the install guide for x86 and x64 architectures for GNU-Linux.

Prolog Mode for Emacs: This is and IDE to manage and debug prolog applications and brings with some tools to understand and read efficiently prolog syntax. Its debug plugin is useful to follow the track of the logic fact deduction.

13 References

- Itself, have. "Have A Function In Scheme Return Value (Or Do Something Else) And Call Itself". Stackoverflow.com. N.p., 2016. Web. 8 Apr. 2016.

- Boyer, C. "Multimagie News." Apr. 4, 2006. `http://www.multimagie.com//English/News0604.htm.`

- Sloane, N. J. A. Sequence A027567 in "The On-Line Encyclopedia of Integer Sequences."

- Rosser, J. B. and Walker, R. J. "The Algebraic Theory of Diabolical Squares." Duke Math. J. 5, 705-728, 1939.

- Kraitchik, M. "Panmagic Squares." §7.9 in Mathematical Recreations. New York: W. W. Norton, pp. 143 and 174-176, 1942.

- Hunter, J. A. H. and Madachy, J. S. "Mystic Arrays." Ch. 3 in Mathematical Diversions. New York: Dover, pp. 24-25, 1975.

- Gardner, M. "Magic Squares and Cubes." Ch. 17 in Time Travel and Other Mathematical Bewilderments. New York: W. H. Freeman, pp. 213-225, 1988.

- Gardner, M. The Second Scientific American Book of Mathematical Puzzles Diversions: A New Selection. New York: Simon and Schuster, pp. 135-137, 1961.