The use of restrictions in Logic Programming: Puzzle Akkoy

Filipa Ramos and Ines Santos

Faculdade de Engenharia da Universidade do Porto,
Rua Dr. Roberto Frias, 4200 - 465, Porto, Portugal
{up201305378,up201303501}@fe.up.pt
FEUP-PLOG,Turma3MIEIC06,GrupoAkkoy_2
https://sigarra.up.pt/feup/pt/web_page.inicial

Abstract. The present report serves the purpose of explaning the process of finding a solution for akkoy puzzles using programming with restrictions. It also refers the implementation of restrictions in order to generate random puzzles. The main objective is to deepen the knowledge of *PROLOG*, specially the clpfd library. The project was developed for the curricular unit of Logic Programming. The results will be evaluated in order to realize the efficiency of the found solution.

Keywords: restrictions programming logic prolog efficiency

1 Introduction

This project was developed for the curricular unit of Logic Programming in order to deepen the knowledge of the clpfd prolog library. Between the many objectives accounted for this project, the following can be highlighted: examining the results of the use of restrictions whilst programming, understanding the logic of rule-based languages, realizing the advantages of logic in programming. The analized puzzle has many restrictions which made it hard to find a solution which incorporated all the restrictions. Besides this, the restrictions must be applied to different objects such as columns, rows and areas. The implemented solution uses the following approach: find out the different possibilities for each column and row according to the numbers given by the problem. The board that fills the restrictions in every row and column will be the solution. The returned board is a list of variables. Each number one represents a black square and each number zero represents a white square. The article is structured in order to make it easier to understand the solution. Firstly, the problem is described. Secondly, the solution and its visualization is explained. Finally, the results are analized and the conclusions are drawn.

2 Problem Description

The presented problem consists in solving an akkoy puzzle. This puzzle has a blank board with numbers on the top and on the right. Figure 1 represents a puzzle with size seven.

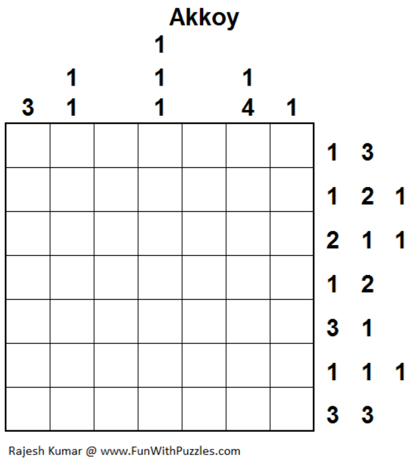

Fig. 1. Example Puzzle.

The numbers on the top represent the number of black squares in each column. For example, in the fourth column there must be three black squares separated. The numbers on the right represent the number of white squares in each line. If these requirements are met, the solution will be a drawing composed of black and white areas (such as Figure 2).

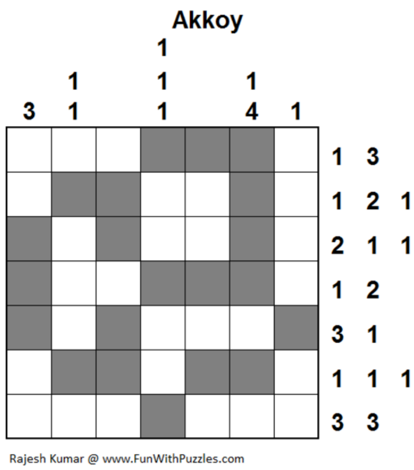

Fig. 2. Solution of the example puzzle.

If there are no numbers in a certain line or column it means that there are no restrictions on the respective line or column (restriction represented by [x]).

3 Approach

3.1 Decision Variables

The following are the several decision variables used in the developed project:

1. `getPossibilities(S, Begins, R1)`
   (a) S represents the size of the row
   (b) Begins are the decision variables
   (c) R1 is the list with the restriction for the column
   (d) domain is between 1 and S

   - Returns a list with the possible index positions for areas in a row according to the given restrictions. The domain is from one to S because it represents all the possible positions on the row.
2. `apply_restrictions(S, List, Restrictions, Color)`
   (a) S is the size of the row
   (b) List is the row that is being restricted
   (c) Restrictions is the list of the numbers for the respective row
   (d) Color is the color that is being analized (white if it is a row/ black if it is a column)
   (e) domain is between 0 and 1

   - Restricts a single row. Domain is between black or white, 1 and 0 respectively.
3. `akkoy(Rcolumns, Rrows, Rows)`
   (a) Rcolumns is the list of lists of restrictions for each column
   (b) Rrows is the list of lists of restrictions for each row
   (c) Rows is the solved board
   (d) domain is between 0 and 1

   - Finds the solution for the problem by applying the restrictions.

3.2 Constraints

The following constraints are applied:

1. *List of numbers* - A number restriction is when there is a number or more on a column or row. These constraints are applied by `apply_restrictions` which summons `apply_merged` and `apply_single_merged`. `apply_restrictions` calls the other two until there are no more rows or columns to restrict. The two paint the cells in the right indexes (found with`getPossibilities`) with the given color and paint the rest of the row the opposite color afterwards.
2. *Empty list* - An empty list means there are no cells with that color on the row. If `apply_restrictions` is called with an empty list of restrictions it calls `swap_color` and `color_all` which color the entire row with the opposite color.
3. *No restriction* - If a row has no restrictions any combination is possible. This is represented by [x]. If this happens, `apply_restrictions` does not have any effect.

3.3 Search Strategy

The labeling is called with an empty list of options (same as [leftmost,step,up,all]). This happens because of the problem which does not need any specific method to restrict the board.

4 Solution Presentation

5 Results

6 Conclusions and Future Work

7 References

References

1. Sicstus Prolog Manual, https://www.kth.se/polopoly_fs/1.339598!/sicstus.pdf
2. Clpfd Documentation, https://sicstus.sics.se/sicstus/docs/4.1.0/html/sicstus/lib_002dclpfd.html

8 Annex

$$\psi(u) = \int_o^T \left[\frac{1}{2} \left(\Lambda_o^{-1} u, u \right) + N^*(-u) \right] dt \, . \tag{1}$$