Katholieke
Universiteit
Leuven

**Department of
Computer Science**

PROJECT

Advanced Programming Languages for A.I. (H02A8a)

HALILOVIC-DERUYTTERE

Armin Halilovic (r0679689)
Thierry Deruyttere (r0660485)

Academic year 2016-2017

Contents

Introduction

In this report we will discuss the different approaches we tried to eventually come to the solutions we have now for Sudoku and Hashiwokakero. The solutions we got are the result of a lot of work and a lot of back tracking on our previously done work. We often came in situations where we got stuck because of the limitations of the ECLiPSe and CHR systems but we also often had to back track on our work since we were often feeling that we were doing things in a non declarative way. We often tried to do things in a procedural way when we first started with Sudoku which means we lost quite some time here since we often had to rethink how we could write things in a more declarative way. For the Hashiwokakero part of the project things went a bit better but we lost quite some time here with the fact that ECLiPSe doesn't support constraints in conditionals. We will discuss this further in chapter 1.1. The fact that we often had to backtrack on our work was according to us due to the fact that we are still novices with prolog since this is the first time we used this programming language.

In our solutions we decided to only use ECLiPSe and CHR. This was partially due to the fact that when we started this assignment we still hadn't seen Jess in class. Once we did have the class about Jess we found that since we were still novices at declarative programming languages, it would be a good exercise to continue using the more declarative systems to gain more experience with them since Jess can also be used to program in a more procedural way. Another reason for not using Jess was that the Jess syntax looked less appealing than the syntax CHR was offering us by all the parenthesizes used in its syntax.

Part 1: Sudoku

1.1   Other Viewpoint

For our other viewpoint for Sudoku we decided that each number has N positions (with N the size of a row of a sudoku board). Each of these (X,Y) positions must be different as well as each X must be different from the other positions as well as the Y. This representation has as a benefit that you know that each value from 1 to N must be in X, so we can already fill this in. The only thing left is to find the right Y.
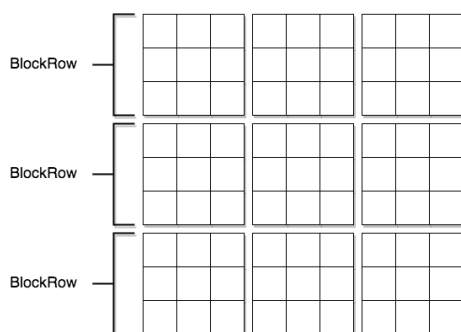

Figure 1.1: Illustration of the definition of a block row

The block constraints can be expressed with a bit of maths thank to this representation. We know each X of all the different positions of a number so we can check in which block row (see illustration) the number is. For example if you take a 9 x 9 board, you know that the rows 1,2,3 belong to block row 1, the rows 4,5,6 belongs to block row 2 and 7,8,9 belongs to block row 3. To express the block constraints we do the following

1. Take all the positions of a number

2. Check which positions are in the same block row (there are exactly $\sqrt{N}$ positions of a number in the same block row).

3. Express that each position in the same block row should be in a different block in that block row.

Part 2: Hashiwokakero

2.1 ECLiPSe

2.1.1 Optimization

Optimization for 4 in corner, 6 at one of the sides or 8 on the board is already fixed implicitly by the (see comments). Single neighbor is also already fixed.

2.2 CHR

2.2.1 Optimization

Conclusion

3.1   Weak points

3.2   Strong points

3.3   Lessons learned

Never use this system ever again.

Appendix

We started working on this project before the easter holiday. During our start we often lost quite some time since we didn't really know how the systems worked. During the second week of the easter holiday we continued to work on the project each evening and we finished the Sudoku part at the end of the holiday. We then had to pause our work since we had an enormous deadline for an other course so sadly enough we could only continue working on the project after this deadline (10th may). As the semester was coming to an end other deadlines also started to come closer and closer so we had to manage these first, so it is only at the start of the study period that we could continue our work. From the start of the study period we tried to spend around 6 hours of work each day for this project. The work was not really divided since we were doing pair programming most of the time. Sometimes someone made some individual changes when they had time but most of our work has been made in group over hangouts by using screensharing. We could argue that by doing pair programming we lost quite some time, which is true, but by doing this we worked very closely together and we learned quite a lot.