APLAI Assignment 2015-2016

February 15, 2016

1 Introduction

The APLAI assignment has 3 parts. To pass for this course you have to do Task 1 and Task 2. Task 3 is optional.

1.1 Some practicalities

- You work in groups of 2: please put your group on the wiki of APLAI (see the Course Documents on Toledo).
- Due date: Monday 06/06/2016 13h00. You email your report (a pdf file), programs and README file to gerda.janssens@cs.kuleuven.be.
- As this assignment is part of the evaluation of this course, the Examination Rules of the KULeuven are applicable. During the exam period, there is an oral discussion for each group on one of the following dates: 15/06/2016, 21/06/2016 and 22/06/2016. Further arrangements via Toledo.
- Follow the rules of academic integrity:
  - Write your own solutions, programs and text. Run your own experiments. When receiving assistance, make sure it consists of general advice that does not cross the boundary into having someone else write the actual code. It is fine to discuss ideas and strategies, but be careful to write your programs on your own. Do not give your code to any other student who asks for it and do not ask anyone for a copy of their code. Similarly, do not discuss algorithmic strategies to such an extent that you end up turning in exactly the same code.
  - Use a scientific approach and mention your sources.
- Some more information about the report:
  - A typical report (for Task 1 and 2) consists of 20 to 25 pages in total (using a font and layout similar to this text). It contains the answers for Tasks 1 and 2, and optionally Task 3.
  - It has an introduction and conclusion. In the conclusion you give a critical reflection on your work. What are the strong points? and the weak points? What are the lessons learned?

– You can include some code fragments in your report to explain your approach. If you do, make a good selection. Do not include the complete code in the report as the code is in the program files.
– When you run your experiments, do not just give the time and/or search results, but try to interpret the results. Include the benchmark problems given on Toledo in your experiments.
– Add an appendix that reports on the workload of the project and on how you divided and allocated the tasks in this project.

1.2 APLAI Systems

The APLAI WIKI page contains information about the installation and use of the ECLiPSe, CHR and Jess systems. THE APLAI WIKI page also contains additional ECLiPSe and CHR exercises.

2 Task 1: Sudoku

2.1 Task 1.A Viewpoints and Programs

The classical viewpoint for Sudoku states that all numbers in a row must be different, that all numbers in a column must be different, and that all numbers in a block must be different.

- Give a **different** viewpoint.
- What are the criteria you would use to judge whether a viewpoint is a good one or not?
- Can you define **channeling** constraints between your alternative and the classical viewpoint? [1]
- For 2 out of the 3 systems (ECLiPSe, CHR and Jess): program the 2 viewpoints and if possible the channeling between the 2 viewpoints.
- Briefly comment on why exactly you did choose the two systems and why you did not use the third one.

Motivate your choices/decisions.

2.2 Task 1.B Experiments

Run experiments with your programs, for the different viewpoints and possibly channeling constraints, and discuss the obtained results (i.e., the run-times and search behaviour (number of backtracks)) in a scientific way.

On Toledo you find a set of Sudoko puzzles (see `sudex_toledo.pl`). You should include them in your experiments, using the same names to refer to the puzzles, but you can also include your own favorite Sudoku puzzles!

[1] Channeling as defined in the APLAI course and also in section 1.9 of `http://www-module.cs.york.ac.uk/copr/handbook-modelling.pdf`.

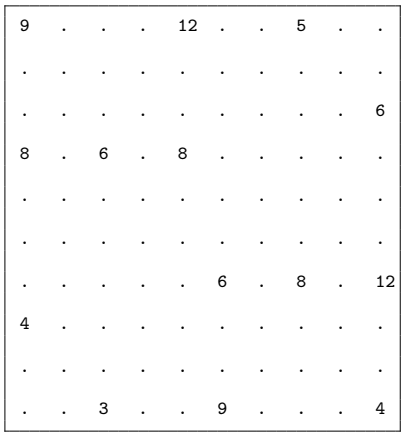

Figure 1: A Shikaku puzzle ...

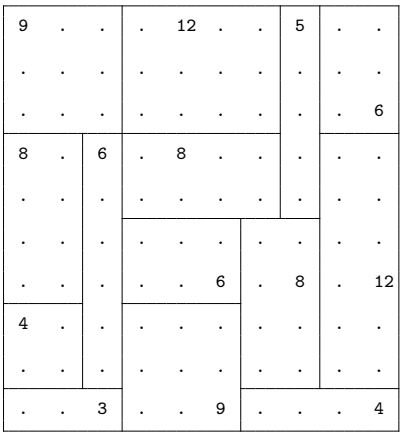

Figure 2: ... and its solution

- Give the results (e.g. timings and number of backtracks) for the (given) Sudoko puzzles and explain them.
- Do you get different results when using different search heuristcs? for your 2 implementations? when using different viewpoints? using channeling? Why?
- What is/are the most difficult puzzle(s) for your programs? Can you explain why this is the case?
- What are your conclusions?

3 Task 2: Shikaku

See also `https://en.wikipedia.org/wiki/Shikaku`.

3.1 Problem definition

A Shikaku puzzle consists of a rectangular grid in which some numbers are placed. The goal of the puzzle is to partition the grid into non-overlapping rectangles, such that each rectangle contains exactly one of the given numbers and the size of each rectangle is equal to the number that it contains. Figures 1 and 2 contain an example of such a puzzle, and its solution.

The task is to write two programs, the first using ECLiPSe and the other using CHR or Jess, that solve instances of these puzzles.

3.2 Example input and output

On Toledo you will find a set of problems (see `puzzles.pl`) generated from the website `http://www.puzzle-shikaku.com`.

Each `problem(Id,W,H,Hints)` fact defines the input of one problem: its identifier `Id`, the width `W` and height `H` of the grid, and the list of initial hints `Hints`. Each hint takes the form `(X, Y, N)` where `X` is the column number, `Y` is the row number and `N` the size of the rectangle surrounding that hint. The origin of the grid starts at the top left corner, the indices start from (1,1).

An example problem and its visualisation is as follows:

```
problem(tiny, 3, 3, [(1,1,4), (3,1,2), (2,3,3)]).
```

4	.	2
.	.	.
.	3	.

The output of the puzzle consists of a set of rectangles. Each rectangle is encoded by a term `rect(ID, Pos, Size)` where `ID` is the coordinate of the hint contained in the rectangle, `Pos` is the coordinate of the top-left corner of the rectangle, and `Size` is the size of the rectangle. The coordinates are encoded as `c(X,Y)` and the size is encoded as `s(Width, Height)`.

The solution of the puzzle is

```
Solution = [
    rect(c(1,1),c(1,1),s(2,2)),
    rect(c(3,1),c(3,1),s(1,2)),
    rect(c(2,3),c(1,3),s(3,1)),
]
```

4	.	2
.	.	.
.	3	.

3.3 Tasks

3.3.1 Task 2.A: Implementation in ECLiPSe

1. Implement in ECLiPSe a **basic solver** that finds a solution for the Shikaku problem as defined above.

   Address the following questions in your report:

   - How do you represent the grid and the rectangles?
   - What are your decision variables?
   - Explain which constraints you use and how they are expressed in your program. Are the constraints active or passive ones?
   - Discuss the impact of the different search strategies.
   - For your experiments, you can use the instances in `puzzles.pl`. The puzzles up to size 10x10 should solve in less than 10 seconds.

2. Propose one additional improvement (e.g. a redundant constraint). Inspiration can be found in the paper "Shikaku as a Constraint Problem"[2]

[2] `http://4c.ucc.ie/~hsimonis/shikaku.pdf`

which gives some useful insights on how to represent and model this problem.

Include this improvement in your solver and discuss its impact.

3.3.2 Task 2.B: Implementation in CHR/Jess

In order to program this problem in CHR or Jess, you will have to encode more things yourself such as constraint propagation and search.

Your report should at least describe the following steps:

1. Choose and explain a suitable representation of the data.
2. Choose and explain a suitable representation of the constraints.
3. Describe how you deal with constraint propagation, and what kind(s) of propagation you support. Are the constraints passive or active? Try to have active constraints if possible.
4. Implement a basic solver that finds a (first) solution. What kind of search is it using?
5. Propose some additional constraints that may speed up solving, and implement a at least 2 of them.
6. Describe the effect of these additional constraints on the performance of your implementation.

If you use CHR for this part, you can have a look at the CHR program for the N-queens problem (given on Toledo). It uses a finite domain solver.

General tips:

- CHR and ECLiPSe are both based on Prolog. You may be able to reuse some code between both tasks.
- For CHR, use the SWI-Prolog version and not the version in ECLiPSe.
- If your CHR program makes SWI-Prolog run out of memory (e.g. Global Stack), restart the SWI system. Also after reloading your CHR program file a number of times, it is a good idea to restart SWI anyway.
- The paper paper "Shikaku as a Constraint Problem"[3] gives some useful insights on how to represent and model this problem.
- On Toledo you can find a Prolog file (`print.pl`) that can be used to visualize the puzzle and its solution for Eclipse and CHR. Once this file is loaded, you can visualize a puzzle by calling the predicate `show(Width, Height, Hints, Rectangles, Format)`. It support three formats: `unicode`, `ascii` and `latex` (for use with the pmboxdraw package).

```
problem(ID, W, H, Hints), show(W, H, Hints, [], unicode).
```

[3] `http://4c.ucc.ie/~hsimonis/shikaku.pdf`

For CHR, you can pass the atom `chr` instead of the list of rectangles. It will then use the `rect/3` constraints from the constraint store.

```
problem(tiny, W, H, Hints),
solve_shikaku(W, H, Hints),    % executes your chr solver
show(W, H, Hints, chr, unicode).
```

- On Toledo you can also find the output (including solutions and execution times for all puzzles) of an implementation in CHR (see `output.txt`).

4 Task 3

There are 3 alternatives for this optional task.

1. Study your own problem in detail.
   - Describe your own problem to be solved.
   - Why did you take this problem?
   - Why do you think it can benefit from the APLAI methods?
   - Write and discuss your 2 programs in detail.
   - Do the APLAI methods work for your problem?
2. There exist other constraint-based systems such as Gecode (http://www.gecode.org/) and MiniZinc (http://www.g12.csse.unimelb.edu.au/minizinc/), ...
   Select one of these alternative constraint-based systems.
   - Suppose you have to choose between the alternative system and the systems studied in this course. Give a detailed overview of the arguments on which your choice is based. Explain your final decision.
   - Write a program in the alternative system to solve Task 2.
3. Identify a related research topic.
   - Make a selection of three to five relevant research papers.
   - Write a short paper (4-5 pages) that motivates the selection of the topic, that summarizes and discusses the papers. The idea is to convince me that this topic should become a part of the APLAI course.