# APLAI Assignment 2012-2013

March 5, 2013

## 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: the 2013 groups are listed on Toledo (wiki of APLAI).
- Due date: Monday 03/06/2013.
- 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 19/06/2013 or on 26/06/2013. If these dates are not possible, 28/06/2013 is scheduled as an additional date. 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:
  - 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?

  - When you run your experiments, do not just give the time and/or search results, but try to interpret the results.
  - 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.
  - Add an appendix that reports on the workload of the project and on how you divided and allocated the tasks in this project.
  - A typical report consists of 20 to 30 pages.

### 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 Discussion

Discuss how Sudoku can be solved by the 3 different systems (ECLiPSe, CHR and Jess). Note that there can be more than one way to solve Sudoku within one system!

Explain in detail how Sudoku can be solved:

- which aspects of the system are useful and why;
- which aspects are not useful and why not;
- possible alternatives, advantages/disadvantages

### 2.2 Task 1.B 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.
- Can you define **channeling** constraints between your alternative and the classical viewpoint?
- What are the criteria you would use to judge whether a viewpoint is a good one or not?
- For 2 out of the 3 systems: program the 2 viewpoints and if possible the channeling between the 2 viewpoints.

Motivate your choices/decisions.

## 2.3 Task 1.C Experiments

Run experiments with your programs, for the different viewpoints and possibly channeling constraints, and discuss the obtained results (e.g. w.r.t. run-times and search behaviour) 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!

- Give the results (e.g. timings and number of backtracks) for the (given) Sudoko puzzles and explain them.
- Do you get different results 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: Timetabling problem: gym

Development of a timetable can be a very difficult practical task. Timetable of a school for example must follow many constraints to satisfy criteria such as the necessary hours of courses being taught and to avoid problems such as demanding professors to give two lectures at the same time or squeezing a group of 100 students into a classroom with 20 seats. All these criteria can be nicely described with constraints and constraint programming is often used in the process of timetable development in practice. Nevertheless, it still remains a daunting task and good solutions are continually sought for.

## 3.1 Problem definition

Our task is one of the more manageable kinds of the timetabling problems: a simple weekly timetable. We are in the role of a gym manager who has to make a weekly timetable for a number of sports. Each sport has a specified number of hours per week. A week starts on Monday and ends on Friday. On any given day, one sport takes place only in a single time slot (we cannot have *sport1* for two hours, then interrupted by an hour of *sport2* and then two hours of *sport1* again). Durations of time slots can be 1, 2 or 3 hours. As the gym manager has to be present while the activities take place in his gym, the solution should be such that on Fridays the activities end as soon as possible.

## 3.2 Example input and output

Program for solving this problem in ECLiPSe would be called with an input of this kind:

```
gym(Timetable, SportsHours, DailyStart, DailyFinish).
```

where the *Timetable* is the solution we are looking for, the *SportsHours* is a list with the number of hours per week that each sport must take, the *DailyStart* is the opening hour of the gym and *DailyFinish* is the closing hour.

An example input for four sports, where the first and the second ones take 4 hours a week, the third takes 2 hours, the fourth takes 8 hours and the gym is open from 16:00 to 20:00 would look like that:

```
gym(Timetable, [4,4,2,8], 16, 20).
```

and the resulting timetable would be represented by a list of daily timetables, where each daily timetable is represented by a list of starting times and a list of durations for all the sports:

```
Timetable = [
[[16, 16, 16, 17], [0, 0, 1, 3]],
[[16, 16, 16, 17], [0, 0, 1, 3]],
[[16, 16, 16, 18], [0, 2, 0, 2]],
[[16, 18, 16, 16], [2, 2, 0, 0]],
[[16, 16, 16, 16], [2, 0, 0, 0]]
].
```

Note that the sessions always start on the hour, that we indicate that a sport is not scheduled on a particular day by setting 16 as its starting time and a 0 duration, and that in this example the last activity is on Friday and ends at 18h00. There exist alternative time tables.

The input of the program can have an arbitrary number of sports and arbitrary opening and closing times. Your program should compute one of the many solutions.

## 3.3 Tasks

### 3.3.1 Task 2.A: Discussion

Discuss how weekly timetabling can be solved by 2 out of the 3 systems studied in APLAI. Explain:

- which constraints need to be implemented;
- which aspects of the systems influenced your choice;
- which ones you selected and why

### 3.3.2 Task 2.B: Implementation in ECLiPSe

Implement a solver for the problem of weekly timetables for a gym as defined above.

As part of the solver, implement your own additional predicate called `no_overlapping(DayStarts, DayDurations)` that prevents overlapping of activities in one day and use it meaningfully in your solver.

Add some symmetry breaking constraints to reduce the number of alternative solutions.

Motivate your decisions (viewpoint, constraints, search strategy, design choices and any decisions for implementing additional predicates).

### 3.3.3 Task 2.C: Alternative representation

Instead of the representation proposed in sections 3.1 and 3.2, use the time slots representation approach taken in this paper:

Timetabling in Constraint Logic Programming; Maria Kambi, David Gilbert; In Proceedings of 9th Symp. on Industrial Applications of PROLOG (INAP'96).

The input format must remain the same, but you can change the output format (the `Timetable` format). If you have changed the output format, describe it.

- Did you find one or the other representation in general more suitable for expressing the problem? Why?
- Did you find some constraints to be much more naturally (easily) expressed in one of the representations? Which ones and why?
- Is one of your two implementations faster that the other? Can you explain why?

### 3.3.4 Task 2.D: Experiments

Select one of your two implementations as a basis for the following experiments and extensions:

- Replace your `no_overlapping/2` predicate with a built-in predicate `disjunctive/2` and compare their performance. Is one of them faster than the other?
- Use the built-in predicate `search/6` meaningfully in your code. Experiment with several options of its use and discuss the results.
- You should include the examples given in `APLAI_task2_examples.ecl`, but you are also encouraged to add new challenging ones, which can be put on the wiki to challenge the other APLAI students. Note that in this file we only give one of the (possibly many) solutions.

### 3.3.5 Task 2.E: Extension

Extend the solver with the option for considering equipment removal time after sports. Namely, after some sports the equipment must be removed, which can cause a loss of time slots before other sports. These losses can even depend on the following sport, as some sports can take place during equipment removal from the previous one.
The input should be extended with another variable called `EqRemoval`, which is a list of equipment removal delays for every sport. However, the equipment removal delay for each sport is again a list where it is indicated for any sport, how long does it take, before it can start to take place in the gym.
For example: $[[0,1,1],[0,0,2],[0,0,0]]$ would indicate that if sport2 or sport3 is scheduled after sport1, there is 1 hour delay ($[0,1,1]$). It also indicates that if sport3 is scheduled after sport2 there would be a 2 hour delay ($[0,0,2]$). Scheduling sport1 after sport2 incurs no delay. Similarly, scheduling sport1 or sport2 after sport3 has no delay either ($[0,0,0]$).

### 3.3.6 Task 2.F: Gym timetabling in CHR/Jess

In order to program this problem in CHR or Jess, you will have to encode more things yourself, as there are no libraries for constraints propagation or search. Start with a basic version for constraint propagation and search. The following issues are important:

- Explain how you model the constraints of 2.A and of 2.B (e.g. the no-overlap constraints).
- How do you deal with constraint propagation? What kind(s) of propagation do you support?
- How do you deal with search? with the minimization?
- Once you have a version that can deal with the simple instances such as the current example, propose two improvements and evaluate them.

Discuss the alternatives you considered and/or implemented and motivate you choices/decisions.

Run experiments with your programs and discuss the results. Having basic versions of constraint propagation and search, your programs might not be able to deal with some of the sample instances. If this is the case, add you own examples/instances that illustrate the strong and/or weak points of your program versions.

**Tips:**

- For CHR, use the SWI-Prolog version and not the version in ECLiPSe predicate.
- 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 file with instances can be loaded into SWI by the Prolog query ?-['APLAI_task2_examples.ecl'].

## 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.