Fundamentals of Artificial Intelligence: Constraint Processing Assignment

In this project you are presented with a number of constraint problems. The goal is to solve these problems using the constraint solver included with *SICStus Prolog*. This solver employs a hybrid algorithm that combines *backtracking* with *arc consistency checking*. Specifically, we discern the following major parts in the constraint solving process: first, an arc consistency algorithm such as AC3 is applied. Then, in case a solution has not yet been found, a loop is entered in which one of the variables is assigned a value from the remaining values in its domain, and the arc consistency algorithm is reapplied. Finally, whenever an inconsistency occurs during this process (i.e., there is no longer a legal value to assign to a particular variable), backtracking takes place over the assignments. The behaviour of backtracking is as described in the course for *forward checking* and *look-ahead checking*.

As opposed to what we considered throughout the course, the solver does not limit the number of variables that are allowed to appear in a constraint. This means that the algorithms employed are not exactly those that were presented in the course, but rather extensions that are capable of handling single constraints that employ more than two variables.

Details on the software that you will use to solve the following problems can be found at: `http://dtai.cs.kuleuven.be/education/ai/Projects/constraints`.

The web page of the constraint processing tool has on the bottom right a field labelled 'Submitted variables & constraints'. Your report should include the content of this field for each of the exercises (ensure that you first clicked on 'Show variables and constraints'). Unless the number of solutions is clearly too great, you should also include the output of the tool (though make sure the 'Debugging' check box is turned off). If there are too many solutions, it suffices to simply mention exactly how many solutions there are.

If the exercise requires you to discuss something, you should ensure that you clearly separate your observations from your explanation of those observations. For every constraint that you use to model each of the problems, your project report should also contain a natural language description of its meaning. For instance, for a constraint in the q-queens puzzle, you could state: 'Two queens should not be on the same column and not on the same diagonal'

Hint: The provided software allows only a single level of indexing. As a result of this, you can not specify a location on a grid using a coordinate system (e.g. in a puzzle refer to square: (4,5)). Instead, you have to assign a unique

0	1	2
3	4	5
6	7	8

Figure 1: A puzzle with single digit indexing

coordinate to each square, as shown in Figure 1.

Using this indexing, the relation between the value associated with a square and its row and column in the puzzle, is determined using the following equations:
$Column(i) = i \mod width(puzzle)$
$Row(i) = \left\lfloor \frac{i}{width(puzzle)} \right\rfloor$
e.g., in the puzzle shown in Figure 1, the square denoted using number 5 would be located in column 2, and row 1.

If there are problems with the constraint tool, or if you feel there are ambiguities in the assignment, please report them to:
dries.vandaele@cs.kuleuven.be
danny.deschreye@cs.kuleuven.be

1 Summation problem

In the following sum, a circle represents an uneven number in the range [1, 6]. A square represents an even number in the same range. The same number can appear multiple types. Use the provided software to determine all the possible ways to fill in the given sum.

$$\begin{array}{rcccc} & & \circ & \square & \square \\ & & \square & \circ & \circ \\ \hline + & \circ & \circ & \square & \circ \end{array}$$

2 Stacking squares

Given a set of squares (2x2, 3x3, 4x4, 5x5), and a rectangular grid (9x7) as shown in Figure 2, place the squares on the grid in such a way that no overlap occurs. It is not allowed to rotate the squares. Additionally, the corners of the squares can only be placed on integer coordinates of the grid (e.g. a corner may have position (1,1) or (1,2), but not something like (1, 1.2)).

1. Use the provided software to find all valid placements of the squares on the grid

2. Assume a solution to this problem contains a 'vertical hole' if it is possible to move at least one of the squares one row downwards while remaining a valid solution. Similarly, assume a solution has a 'horizontal hole' when

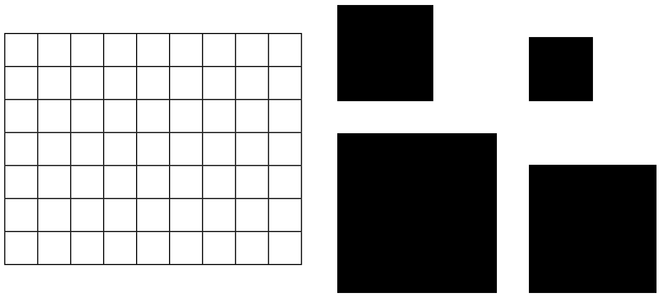

Figure 2: Fill the grid with the given squares

it is possible to move a square one column to the left while ensuring that there is still no overlap. Now modify your model to generate only those solutions that do not contain vertical or horizontal holes.

3 Docked ships

A number of ships need to dock in a harbour for the purpose of loading and releasing cargo. The dock has been split up in several areas. An area can be assigned to a ship for several hours. Some ships are so large that they require multiple adjacent areas in order to successfully dock. The goal is to determine if it is possible (and in case it is possible: how?) to assign the various areas during certain times to specific ships, in such a way that each ship manages to successfully dock, and has its requirements satisfied. Concretely we consider the following ships:

- Ship 1 has a length of 4 areas and has to be docked for a period of 3 hours.
- Ship 2 has a length of 2 areas and has to be docked for a period of 2 hours.
- Ship 3 has a length of 3 areas and has to be docked for a period of 1 hour.
- Ship 4 has a length of 5 areas and has to be docked for a period of 1 hour.

The dock contains a total of 7 areas and is available for a total of 4 hours.

This problem can be considered a variant of the problem in Section 2, where rectangles have to be assigned instead of squares. Discuss the exact relation between these problems. Then, modify your constraints for the exercise in Section 2 to solve this problem. As with the previous exercises, the resulting solution has to be included in your report.

The constraint processing tool offers the possibility of experimenting with different strategies for deciding the order in which variables are selected. What do you observe when you consider the required number of backtracks to find solutions using the simple *leftmost* heuristic? What happens when you instead choose to use the *first-fail* heuristic? Explain!