Assignment 2: CSPs

CSC 384H—Fall 2015

Out: Oct 13th, 2015
Due: Electronic Submission Tuesday Oct 27th, 7:00pm
Late assignments will not be accepted without medical excuse
Worth 10% of your final mark.

Be sure to include your name and student number as a comment in all submitted documents.

Handing in this Assignment

What to hand in on paper: No paper submission required.

What to hand in electronically: You must submit all your answers and code electronically. You must submit the following:

1. A file of python code called `propagators.py`. This file will contain your implementation of forward checking and GAC. The routines `prop_FC` and `prop_GAC` can be used to do constraint propagation within the `bt_search` routine.
2. A file of python code called `sudoku_csp.py`. This file will contain two functions that build a two different CSP models (returned as a `CSP` object) of an input Sudoku problem. These CSP models can then be solved with the `bt_search` routine to obtain a solution.

To submit these files electronically, use the CDF secure Web site
`https://www.cdf.toronto.edu/students/`
or use the CDF **submit** command. Type **man submit** for more information. The name of the assignment for submit will be "A2"

Since we will test your code electronically, you must

- *make certain that your code runs on CDF using python3 (version 3.4.1* (installed as "python3" on CDF).
- not add any non-standard pythons imports from within the python files you submit (the imports that are already in the template files must remain).

Introduction

In this assignment you will implement two constraint propagtors. `prop_FC` is a procedure that given an CSP (with constraints and variables and where some variables might be assigned) computes all variable value pairs that would be pruned by the forward checking algorithm. Your `prop_FC` procedure will also passed the variable that has been newly assigned.

The other propagator `prop_GAC` will implement GAC propagation.

In addition to these two propagators you will be asked to implement two different CSPs models for solving the Sudoku problem. In one model only binary not equal constraints will be used, while in the other model 9-ary all different constraints will be used.

What is supplied. You will be supplied with python code implementing `Constraint`, `Variable` and `BT` objects. The file `cspbase.py` contains the class definitions for these objects. The code supports representing constraints as a collection of satisfying tuples—so to specify a constraint in one has to construct the list of satisfying tuples and pass them to the constraint object.

Note that this representation can be space expensive, especially for constraints over many variables, e.g., those contained in the second Sudoku CSP model.

You will be supplied with two template files `propagators.py` and `sudoku_csp.py` which you will complete with your implementation.

Propagators 50/100 marks

See the files `cspbase.py` and `propagators.py` for the input output specification of the two functions you are to implement.

The correct implemenation of each function is worth 25/100 marks.

To Submit

1. Submit your python implementation in the file `propagators.py`

Question 2. Implement Model 1: worth 50/100 marks

Implement the functions `sudoku_csp_model_1` and `sudoku_csp_model_2`. These two functions take as input an initial Sudoku board, and construct and return a CSP model where for model 1 all constraints are binary not-equals constraints and for model 2 all constraints are larger arity all-diff constraints. A variable is defined for each cell of the board and a matrix of the variables is also returned by your routine.

See the file `sudoku_csp.py` for the detailed specification of these functions.

To Submit

1. Submit your python implementation in the file `sudoku_csp.py`