Artificial Intelligence 1 – WS 2020/2021
Assignment 5: Constraint Propagation
– Given Dec 10, Due Dec 20 –

Problem 5.1 (Definitions) 30 pt

We consider binary CSPs with

- a list V of variables $v_1, \ldots, v_n$
- a family D of domains D_i for $i = 1, \ldots, n$
- a family C of constraints $C_{ij} \subseteq D_i \times D_j$ for $i, j = 1, \ldots, n$, $i \neq j$ where C_{ij} is the dual of C_{ji}

Note that it is easy to assume that C_{ij} are given for all i, j — if we want to omit a constraint, we can simply assume $C_{ij} = D_i \times D_j$ or $C_{ij} =$ "`true`" (all pairs are allowed, i.e., no constraint). That could be problematic in implementations, but is practical on paper.

Assume two CSPs $\gamma = (V, D, C)$ and $\gamma' = (V, D', C')$.

1. Give a definition for $\gamma \sqsubseteq \gamma'$.
2. Give a definition for $\gamma \equiv \gamma'$.
3. Judge the following statement: If $\gamma \sqsubseteq \gamma'$ and $\gamma' \sqsubseteq \gamma$, then $\gamma \equiv \gamma'$.
4. Judge the following statement: If $\gamma \equiv \gamma'$, then $\gamma \sqsubseteq \gamma'$.

Here "judging" means to argue why the statement is true or to give an example that shows that it is false.

Problem 5.2 (Scheduling CS Classes with Constraint Propagation) 40 pt

Consider the CSP problem for scheduling CS classes from the previous assignment.

1. Show the CSP obtained by running arc-consistency. As usual, you can visualize this as a graph whose
   - nodes are labeled with the variable names and domains
   - edges are labeled with the constraints.
2. Give all optimal cutsets for the CSP.

Problem 5.3 (Finite CSPs in Prolog) 30 pt

Consider the Prolog framework for binary CSPs from the previous assignment. (For this exercise, only the provided code is relevant. You will not build on your solutions.)

Implement the following Prolog predicates:

```
% To represent problems with tightened domains, we extend our code as follows:
% A list [L_1,...,L_n] of lists represents a CSP with tightened domains.
% (We can think of the L_i as unary constraints by the way.)
% Each L_i is a list of values from D_i, and only those values are legal.
```

```
% If Ds = [l_1,...,l_n], then initially, L_i=[0,...,l_i - 1].
% When tightening the domains, we remove values from L_i.

% Let Ds, Ls, Cs be such a CSP.
% arcconsOne(Ds,Ls,Cs,I,J) holds if x_I is arc-consistent relative to x_J
% i.e., for every value for x_I from L_I there is a value for x_J from L_J
% such that C_IJ (if present) is satisfied
arcconsOne(Ds,Ls,Cs,I,J) :- ???

% arcconsAll(Ds,Ls,Cs) holds if the whole CSP is arc-consistent
arcconsAll(Ds,Ls,Cs) :- ???

% Given an untightened CSP Ds,Cs as before, makeArccons(Ds,Cs,Ls) returns Ls
% such that the CSP is arc-consistent.
makeArccons(Ds,Cs,Ls) :- ???
```