Fall 2003 **B.Y. Choueiry**

CSCE 421/821: Foundations of Constraint Processing

Homework 4

Assigned: Friday, Nov 7, 2003

Due: Friday, Nov 21, 2003, 10:30 am (at the start of class) Homework 4 must be printed and physically handed out to the instructor.

Total value: 75 points,

Bonus: Question 4 (c), 25 points.

Notes: This homework must be done individually. Please avoid discussing it with classmates, TA or instructor. *If you receive help from anyone, you must clearly acknowledge it.* Always acknowledge sources of information (URL, book, class notes, etc.). Please inform instructor quickly about typos or other errors.

These exercises are borrowed from colleagues, world-wide.

---

1. A simple resource allocation problem **(Total 20 points)**

Consider the following train scheduling problem, where the task is to allocate locomotives to trains. There are 4 trains: T_1, T_2, T_3, and T_4, and 3 locomotives: L_1, L_2, and L_3. The following table shows the schedule for each of the trains:

Train	In use
T_1	8 a.m.-10 a.m.
T_2	9 a.m.-1 p.m.
T_3	noon-2 p.m.
T_4	11 a.m.-3 p.m.

In addition, there are the following problem assumptions:

(a) Each train must be pulled by a locomotive.

(b) Each locomotive can pull only one train at a time.

(c) If a locomotive is not in use, it can immediately move to any station for use by any train.

(d) L_3 is too small to pull T_3.

(e) L_2 and L_3 are too small to pull T_4.

Model this problem as a CSP and solve it using backtrack search with forward checking.

- State the variables, their domains, and the constraints. (5 points)
- Draw the constraint graph. (3 points)
- Assuming the variable ordering T_1, T_2, T_3, and T_4 and the value ordering L_1, L_2, L_3, find all the solutions to this problem using backtrack search with forward checking, showing the search tree as each solution is found. (12 points)

2. A simple application of arc-consistency (10 points)

Consider the following CSP.

- $\mathcal{P} = \mathcal{V}, \mathcal{D}, \mathcal{C}\}$
- $\mathcal{V} = \text{foo}, \text{bar}, \text{baz}, \text{qux}$
- $\mathcal{D} = \{D_{x,\forall x \in \mathcal{V}}\}$ and $D_x = \{1, 2, 3, 4, 5\}$
- $\mathcal{C} = C_1, C_2, C_3, C_4, C_5$ with
  (a) foo = bar +1
  (b) bar = qux
  (c) baz > bar
  (d) qux ∈ 1, 3
  (e) baz = 2 × bar

Draw the constraint network of this CSP. Make the problem arc-consistent, showing each domain reduction along with the reasons why it occurs.

3. A simple application of path-consistency (20 points)

Consider a CSP with the three variables x, y, and z, each with the domain {1, 2, 3, 4, 5}, and the following three symmetric constraints:

$$C_{x,y} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad C_{y,z} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad C_{x,z} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Define the domains of the variables as vectors. Give the six constraints (C_x, C_y, C_z, $C_{x,y}$, $C_{y,z}$, $C_{x,z}$) that result from applying `PC-1` to the CSP. Follow `PC-1` as closely as possible and document the application of each constraint-filtering operation you apply.

4. A crypto-arithmetic puzzle (Total 25 points. Bonus 25 points)

Consider the crypto-arithmetic puzzle shown below.

- You are asked to replace each letter by a *different* digit from 0 to 9.
- No leading zeros are allowed.
- When each letter is replaced by the appropriate number, this cryptogram represents a correct addition problem:

$$\begin{array}{cccccc} & \mathbf{S} & \mathbf{E} & \mathbf{N} & \mathbf{D} \\ + & \mathbf{M} & \mathbf{O} & \mathbf{R} & \mathbf{E} \\ \hline \mathbf{M} & \mathbf{O} & \mathbf{N} & \mathbf{E} & \mathbf{Y} \end{array}$$

(a) *Model this puzzle as a CSP.* (20 points)

List the variables, their domains, and the constraints. Specify the constraint definitions in intension. [Hints: (1) In addition to each letter being a variable, you need to account for the carries. (2) The CSP is not necessarily binary.]

(b) *Draw the constraint network of your model.* (5 points)

Label the nodes with the domains of the variables (in extension), and the constraints with their definitions (in intension).

(c) *Solve the puzzle.* (Bonus: 25 points)

Depending on how you model the puzzle, you may be able to solve it with simple consistency checking on likely non-binary constraints (e.g., node, relational arc-consistency, and relational path-consistency) or may need to simulate a backtrack search. The former may be useful when your model has several constraints of arity 5 or less, the latter may be necessary when your model has one constraint of large arity. (We advise you to adopt the former approach.)

- If your model lends itself to consistency checking, then step through a consistency checking process showing which constraint is checked at each step and how the domains of the applicable variables are updated. Keep in mind that you do not need to visit all the constraints once before you can 'come back' to a given constraint.
- Otherwise, show how you can solve this problem with a backtrack search with a full look-ahead technique (i.e., applying the strategy known as Maintaining Arc-Consistency).

Relational consistency

It is useful to understand the relational-consistency methods in order to solve the puzzle using constraint propagation only (i.e., without search). Refer to Chapter 8 of Dechter's textbook to learn the details of advanced consistency concepts and methods. In class we discussed arc-consistency and path-consistency in great detail in the context of binary constraints. In the context of non-binary constraints, these concepts need to be generalized to relational arc-consistency (a.k.a. generalized arc-consistency) and relational path-consistency. Below is a summary of the class discussion.

The goal of arc-consistency is to update the domain of each of the variables to which it applies. For non-binary constraints, relational arc-consistency can be achieved by direct application of the Waltz algorithm (see Lecture slides 4). The two procedures of the Waltz algorithms `Revise` and `Refine` ensure that, for a given constraint, every value of the domain of every variable is supported according to the constraint by some values in the domains of the remaining variables. For example, in Fig 1, if the constraint C_x is defined such as $V_1+2V_2+V+3 = V_4$, then we consider all combinations of tuples for these variables, remove the combinations that do not satisfy the constraints, then remove from the domains of the variables those values that do not appear in any acceptable combination.

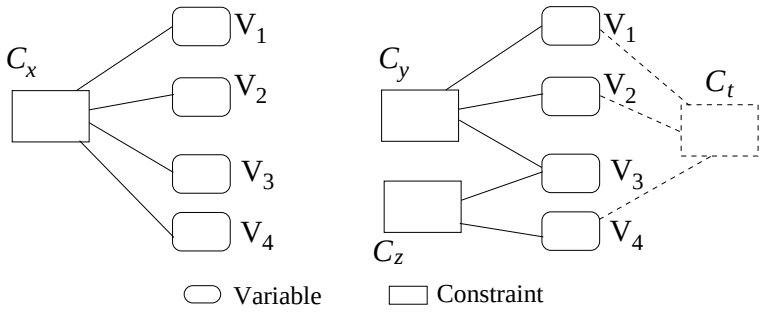

Figure 1: *Left:* Relational arc-consistency. *Right:* Relational path-consistency.

The goal of path consistency is to combine two constraints C_y an C_z and induce a new constraint C_t between the variables that are in the union of the scopes of the original two constraints (i.e., scope$(C_y) \cup$ scope(C_z)) but not in their intersection (i.e., scope$(C_y) \cap$ scope(C_z)). In the example of Figure 1, the scope of C_t is $\{V_1, V_2, V_4\}$. In the case of algebraic expression, this can be achieved by simple elimination. For example, the two constraints:

$$C_y : \quad V_1 + V_2 \quad > V_3 \tag{1}$$
$$C_z : \quad V_3 + 10 \quad > V_4 \tag{2}$$

yield the following induced constraint

$$C_t : \quad V_1 + V_2 + 10 \quad > V_4. \tag{3}$$