Toby Chappell

CPSC 390

**Homework #3 Constraint Propagation:**

**I. (5 points)** In this map-coloring problem, each variable has the domain {red, green, blue}.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **D** |
| **C** |

We start out by coloring country A red.

1. What will the domains of B, C, and D be after doing forward checking? After AC3? [It is not necessary to show the full details of either algorithm. It is sufficient that you give a high-level explanation of why a value is removed from a particular variable’s domain.]

Forward Checking:

1. A - {red}, B - {green, blue}, C - {green, blue}, D - {red, green, blue}
2. A - {red}, B - {green}, C - {blue}, D - {red, blue}
3. A - {red}, B - {green}, C - {blue}, D - {red}

* Forward checking removes the color from the domains of countries adjacent to the country being assigned to.

AC3:

1. A - {red}, B - {green, blue}, C - {green, blue}, D - {red, green, blue}
2. A - {red}, B - {green}, C - {blue}, D - {red}

* AC3 affects the domains of countries not only adjacent to the country being assigned to, but by the countries adjacent to those countries as well. This only occurs when an adjacent country is assigned to by eliminating other variables.

2. Without instantiating any additional variables, what further inconsistent values can you eliminate? Explain why forward checking and AC3 weren’t able to do this.

A - {red}, B - {green, blue}, C - {green, blue}, D - {red}

* D must be red because B and C have to be either green or blue
* Forward checking isn’t able to do this since D does not share a border with A
* AC3 isn’t able to do this since the B and C are not assigned to yet.

3. Formulate a constraint propagation algorithm specific to the map-coloring problem that will correctly detect the kind of inconsistency shown in part 2. Note that this algorithm does not need to duplicate the behavior of forward checking or AC3. Pseudo-code is not necessary, but your formulation should be precise.

Variables: A - {red, green, blue}, B - {red, green, blue}, C - {red, green, blue}, D - {red, green, blue}

Constraints: A != B, A != C, B != C, A = D

1. A - {red}, B - {green, blue}, C - {green, blue}, D - {red}
2. A - {red}, B - {green}, C - {blue}, D - {red}

**II. (5 points)** A machine shop has four jobs to complete – j1, j2, j3, j4. Each one takes an hour and no two can be done simultaneously. Moreover, job one must be completed before job four, and job two after job three.

1. Formulate this scheduling problem as Constraint Satisfaction Problem. [Your answer should start like this: There are 4 variables, say, j1, j2, j3, j4. The domain of j1 is {1, 2, 3, 4} << FILL THE REST >>.]

Variables: j1 - {1, 2, 3, 4}, j2 - {1, 2, 3, 4}, j3 - {1, 2, 3, 4}, j4 - {1, 2, 3, 4}

Constraints: j1 != j2, j1 != j3, j1 < j4, j2 > j3, j2 != j4, j3 != j4

1. Use backtracking search to find a solution. Show the sequence of variable values evaluated and backtracks. [Here, your answer should start like this:  
   Denoting backtrack points (i.e., dead-ends) by BC, the backtracking search produces this sequence:

j1 = 1, j2 =1, BC,

j2 = 2, j3 = 1, BC,

j2 = 3, j3 = 1, BC,

j3 = 2, j4 = 1, BC,

j4 = 2, BC,

j4 = 3, BC,

j4 = 4

Solution Found: j1 - {1}, j2 - {3}, j3 - {2}, j4 - {4}

1. Use local search with min-conflicts heuristic to find a solution. Start with (2,2,2,2) allocation (i.e., j1, j2, j3, j4 = 2) and show all steps. [Instead of picking conflicted variables at random, pick them in their natural order, i.e., j1, j2, j3, j4.]

j1 - {2}, j2 - {2}, j3 - {2}, j4 - {2}

j1 - {1}, j2 - {2}, j3 - {2}, j4 - {2}

j1 - {1}, j2 - {3}, j3 - {2}, j4 - {2}

j1 - {1}, j2 - {3}, j3 - {2}, j4 - {2}

j1 - {1}, j2 - {3}, j3 - {2}, j4 - {4}

1. I now advise you to draw the constraint graph to help you answer the following question, but you don’t have to submit this graph in your answer. Find a solution by interweaving AC3, backtracking and forward checking. Explain concisely what variable values are eliminated or selected at each step.

j1 - {1, 2, 3, 4}, j2 - {1, 2, 3, 4}, j3 - {1, 2, 3, 4}, j4 - {1, 2, 3, 4}

j1 - {1}, j2 - {2, 3, 4}, j3 - {2, 3, 4}, j4 - {2, 3, 4}

j1 - {1}, j2 - {2}, j3 - {}, j4 - {3, 4}

j1 - {1}, j2 - {3}, j3 - {2}, j4 - {4}