600.432/632 Declarative Methods - Prof. Jason Eisner
Answers to "CSP Solvers" discussion problems

1. - The Murderer-Weapon constraint eliminates Prof. Plum, since he has no possible weapon. It may help to cross off all rows involving Prof. Plum in all of the constraint tables.

   - Now that we know that Prof. Plum is out, the Murderer-Room constraint eliminates the Billiard Room.

   - Now that we know that the Billiard room is out, the Weapon-Room constraint eliminates the Lead Pipe.

   - Now that we know that the Lead Pipe is out, the Murderer-Weopan constraint exonerates Mrs. Peacock.

   So the final solution is

Murderer	Weapon	Room
Miss Scarlet	Rope	Study
Colonel Mustard	Revolver	Kitchen
	Candlestick	

   In fact, these constraints do carry enough information to rule out Col. Mustard, since none of his remaining weapons could be used in the Study. (In fact, the constraints imply that it was Miss Scarlet and she used the Rope.)

   But arc consistency applied to these contraints (without joining them) isn't powerful enough to do that. We have to actually try guessing Col. Mustard, discover the contradiction, and backtrack.

2. (a) Nothing happens. Even if Z is restricted in this way, there are still infinitely many possible values for X and also for Y.

   (b) To
          "Project the result onto Y and Z"
       means
          "To eliminate X from the constraint"

       So we proceed with standard elimination:
       - Multiply the first constraint by -2
          - 2X - 2Y - 2Z #< -6
       - Add to the second constraint and get
          Y + 2Z #< 14

   (c) The list of constraints is:

```
X + Y + Z #> 3
2X + 3Y + 4Z #< 20
Z #>= 0
Z #< 2
Y + 2Z #< 14
```

   -   From

```
Z #>= 0
Y + 2Z #< 14
```

       we infer

```
Y #< 14 (i.e., Y #=< 13)
```

   -   Now, from

```
Y #=< 13
Z #=< 2
```

       and the original constraint

```
          X + Y + Z #> 3
      we conclude
          X #> -12 (i.e., X #>= -11)

  -   Now, from
          X #>= -11
          Z #>= 0
      and the original constraint
          2X + 3Y + 4Z #< 20
      we conclude
          3Y #< 42
          Y #< 14 (i.e., Y #=< 13)

      However, we already knew that, so there is no further change
      to Y's domain that needs to be
      propagated.  We are done.

  So the new domains are:
      X :: -11..infinity
      Y :: -infinity..13
      Z :: 0..2.

3.
  (a) -   Eliminate B by combining all equations that mention B:
          A * 2 #= B
          B * 2 #= C
      which yield
          A * 4 #= C,         C
          C * 2 #= D,       / |
          A + D #= 27,      | D
                            \ |
                              A

  -   Eliminate C by ombining all equations that mention C:
          A * 4 #= C
          C * 2 #= D
      which yield
          A * 8 #= D,         D
          A + D #= 27,        |
                              A

  -   Eliminate D by combining all equations that mention D:
          A * 8 #= D
          A + D #= 27
      which yield
          9 * A #= 27.        A

  Notice that the original constraint graph was a cycle.
  Each time we eliminate a node and connect its neighbors
  to each other, we get a new (smaller) cycle.

  To find a complete solution, we can "back-solve."  Here's how it
  works in detail:

  First, find a satisfying assignment for A in the last program: A=3.

  With this assignment for A, we know it is possible to satisfy
  the original constraints, so we move back to the previous program
  that relates D and A.  Setting A=3, we find D=24 (from either equation).

  With these assignments for A and D, we know it is possible to
  satisfy the original constraints, so we move back to the previous
  program that relates C, D, and A.  Setting D=24 and A=3, we can
  find C=12 (from either equation involving C).
```

Finally, we find B=6 similarly.

(b) Eliminating B, we connect B's neighbors (C,D,A) to one another. The result in this case is a complete graph.

In general, this leads to a joint 3-way constraint on C,D,A -- they must have values that are simultaneously compatible with some value of B.

(Despite this 3-way constraint, the conventional way of drawing a constraint graph is only to connect any TWO variables that appear in the same constraint. It turns out that this is enough to pick a variable elimination order, etc.)

4. (a) MAY.

The new constraint is equivalent to the conjunction of the old ones -- that is what "joining" constraints means! So the old constraints are now redundant. Notice C_23 is a constraint over ALL the variables mentioned in C_2 and C_3; we didn't project any out the way that we projected out A in question 3.

(b) i. Could be immediate (if X and Y come first in the variable ordering and we do short-circuit evaluation, we can detect the contradiction immediately by trying each legal (X,Y) pair without having to continue on and assign any of the other variables).

Could also take exponential time, if X and Y come last in the variable ordering.

ii. Immediate only. Arc consistency with the (real) domain of X will eliminate all values < 1 from the domain of Y, and the unary constraint on Y has already eliminated all values >= 0, so Y's domain is empty before we even make any assignments.

iii. Immediate only. Bounds consistency will find as above that Y >= 1 and Y < 0.

(c) (k+1)-consistency, or strong (k+1)-consistency.

This generalization of arc consistency ensures that for any set of k+1 variables, we can find a satisfying assignment for the subproblem that consists of only those variables and only the constraints that involve only those variables. In fact, it does more -- it removes values from the domains of these variables if they do not participate in any satisfying assignment to the subproblem.

So if there is a (k+1)-clique, there are no satisfying assignments at all to the sub-problem of coloring just those k+1 variables, and all values are removed from the domains of those variables, allowing us to detect UNSAT immediately.

The technique is slow because for each set of k+1 variables, one must join the constraints on those variables.