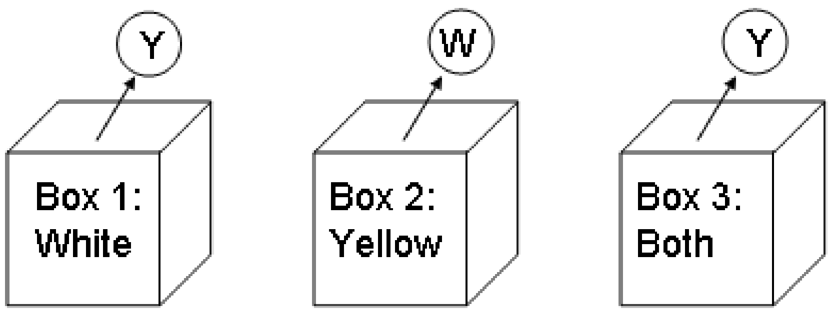
CSCE 625, HW #1
due: Tues, Oct 31, 20167 (hand-in print-out in class - not hand-written)

1. You are the proprietor of Sammy's Sport Shop. You have just received a shipment of three boxes filled with tennis balls. One box contains only yellow tennis balls, one box contains only white tennis balls, and one contains both yellow and white tennis balls. You would like to stock the tennis balls in appropriate places on your shelves. Unfortunately, the boxes have been labeled incorrectly; the manufacturer tells you that you have exactly one box of each, but that each box is definitely labeled wrong. One ball is drawn from each box and observed (assumed to be correct). Given the initial (incorrect) labeling of the boxes above, and the three observations, use Propositional Logic to derive the correct labeling of the middle box.



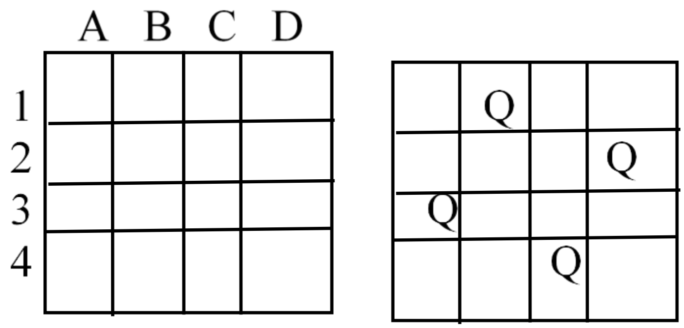
- Use propositional symbols in the following form: O1Y means a yellow ball was drawn (observed) from box 1, L1W means box 1 was initially labeled white, and C1B means box 1 actually contains both types of tennis balls.
- The initial facts describing this particular situation are: {O1Y, O2W, O3Y, L1W, L2Y, L3B}

a. Using these symbols, write a propositional knowledge base that captures the implications of what different observations or labels mean, as well as constraints inherent in this problem (e.g. all boxes have different contents). Do it in a complete and general way (writing down all the rules and constraints for this domain, not just the ones needed to make the specific inference about the middle box). *Do not include derived knowledge* that depends on the particular labeling of this instance shown above (e.g. ¬C1W).

b. Prove that box 2 must contain white balls (C2W) using Natural Deduction.

c. Prove C2W using Resolution Refutation.

2. Write a propositional knowledge base describing the 4-Queens problem.



a. Solve the puzzle by deriving a model using DPLL **using NO heuristics**.
(Trace the steps, indicate what decisions are made in each iteration, indicate if and when back-tracking occurs, and indicate what the final solution is.)

b. "Solve" the puzzle by computing a model using DPLL **using the PureSymbol and UnitClause heuristics**.
(Trace the steps, indicate what decisions are made in each iteration, indicate if and when each heuristic is applied, and indicate what the final solution is.)

3. Write a propositional knowledge base for playing tic-tac-toe. The rules are: players alternate placing pieces (X or O) on a 3x3 board; a win is defined as when a player has 3 of their pieces in a row, column, or diagonal.

Suppose we use a propositional encoding of the game state as follows: Xij means there is an X in row i, column j, Oij is for O pieces, and ?ij indicates a blank position. For example, the board below is described by the following propositional sentence: {X11∧?12∧?13∧O21∧?22∧?23∧O31∧?32∧X33} (top-left is 1,1)

```
X |   |
---------
O |   |
---------
O |   | X
```

The goal is to write rules in propositional logic for where player 1 (the computer, assumed to be X) should place its next piece. In the board shown above, the best move would be in the middle (2,2), which we can represent as a proposition **moveX22**. The reason this is the optimal move is that X can win (**canWinX22**). In other situations, X might not be able to win, but is forced to make a move to block a potential win by O, as in the following (**forcedMoveX32**). Note that you only need to make a forced move if there is no way X can win outright.

```
X |   |
---------
  |   | X
---------
O |   | O
```

Write rules to infer all desirable moves (for X) based on the above strategies. Hint: you might want to define rules for when placing a piece (X or O) in each postion (i,j) would be a win for that player, and then use this to build up more general rules like for canWinX, canWinO, forcedMoveX, etc. Note that there might be multiple optimal moves in a given situation (in which case, one could be chosen at random; this does not need to be encoded in your rules; see example below).

(For a complete tic-tac-toe-playing program, there would be other cases you would need to handle as well, such as defining opening moves, and what to do if there are no positions where canWinX or forcedMoveX. An intelligent player-program might attempt to place two pieces in a row to setup a future situation where there will be two ways to win and the opponent can only block one. You might think about how to write rules for such sophisticated reasoning. However, they are not required for this homework.)

Here are some more examples:

In this case, only **moveX23** should be inferrable (because it is forced).

```
X |   | O
---------
O |   |
---------
  | X | O
```

In this case, only **moveX12** should be inferrable (because X can win).

```
X |   | X
---------
  |   |
---------
O |   | O
```

In this case, you don't have to infer anything, because neither player can win.

```
  |   | X
---------
  |   |
---------
O |   |
```

In this case, it is OK if both **moveX12** and **moveX23** are inferrable.

```
X |   | X
---------
  | O |
---------
O | O | X
```