COMP 424 - Assignment 2

Karla Gonzalez

260715039

1 Question 1: Constraint Satisfaction

Consider the problem of placing two knights and two queens on a 4x4 chess board such that no two pieces attach each other. Suppose that rows 1 and 2 must contain a knight and rows 3 and 4 must contain a queen.

a) Formulate this problem as a CSP with binary constraints. List the variables, their domains and the constraints. Draw the constraint graph

Recall that a CSP is defined by:

1. Set of variables V_i that can take values from a domain D_i - the domains need not be the same for each variable.
2. Set of constraints specifying what combinations of values are allowed. These can be represented implicitly (as functions) or explicitly (as a list of allowable values).

We obtain a solution when we reach an assignment of variables such that all the constraints are true. Let us observe the current problem. We have:

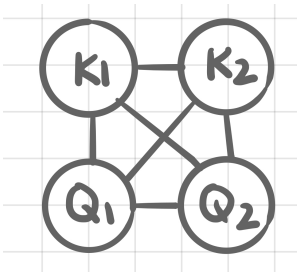
Variables $V_i = K_1, K_2, Q_1, Q_2$ enumerates the ith piece on the board starting with the 2 knights and then the 2 queens.

Domain $D(V_i) = 1,2,3,4$ The values that a variable can have represent the column of where we can place the ith variable (queen or knight).

Constraints = We know that any piece cannot share a column, a row or a diagonal with a queen else it would be attacked. A knight attacks in a different manner that we must also consider. Formulating this numerically would lead to the following:

For any $V_i = (x_i, y_i)$ and $V_j = (x_j, y_j)$ we have that :

- $(x_i, y_i) \notin ((x_j \pm 2, y_i \pm 1)$ for i≠j (cannot be attacked by a night, only applies when j is a knight.)
- $(x_i, y_i) \notin (x_j+1, y_j+1), (x_j-1, y_j-1)$ (not on the same diagonal, this applies when j is a queen).
- $y_i \neq y_j$ for i≠ j (cannot be in the same column, this applies when j is a queen)



b) Run AC-3 on this problem then apply backtracking search to find a possible solution

Let us remind ourselves what the AC-3 algorithm does. Let us look at the following pseudo code:

AC-3 (CSP):

WL ← all arcs in the CSP's constraint graph while worklist (WL) is not empty:

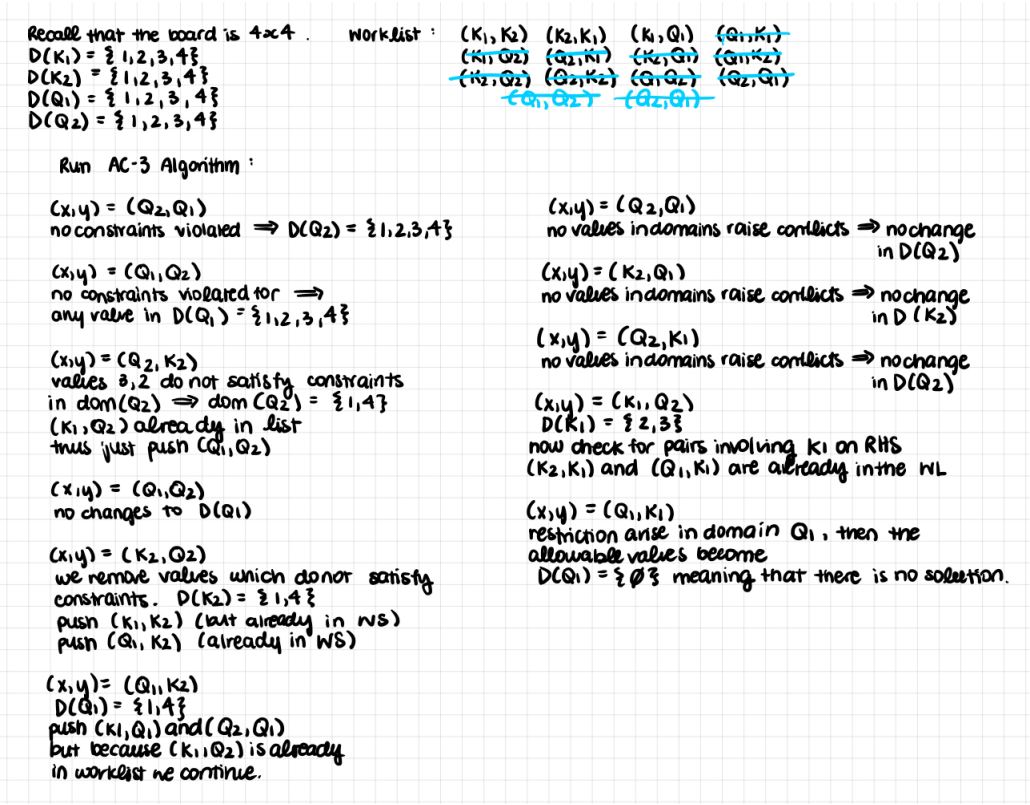
(X,Y) ← WL.pop()

prune dom(X) where constraints involve Y

if size(dom(X) ==0): no solution.

if dom(X) was changed: for neighbours of X except Y: WL.push((N,X))

run search algorithm if necessary



c) Starting from the original problem (without running AC-3) run backtracking search with one step forward checking.

Let us recall Backtracking Search Algorithm: (how to pick a remaining unassigned variable at each level of a search tree.)

- select an unassigned variable , X
- for each value = $\{x_1, ..., x_n\}$ in the domain of that variable.
  - If the value satisfies the constraints, let X=xi and exit loop
- if an assignment was found, move onto the next variable.
- If no assignment, go back to the preceding variable and try a different value

	Nothing Assigned	Assign K1 = 1	Assign Q1 = 4
K1	1,2,3,4	1	1
K2	1,2,3,4	1,2,4	1
Q1	1,2,3,4	4	4
Q2	1,2,3,4	2,4	Ø

We assign Q1=4 because that is the only available number for that domain after we assign K1=1. This then leads to an automatic assignment of Q2=2, again because it was the only value left in its domain. But this is wrong because in this configuration we get that K2 is attacking Q2 and that is not what we want to we need to go back retry another value for K1. We can see that K1=2 would immediately eliminate all possible values for Q1. This is shown below.

	Nothing Assigned	Assign K1 = 2
K1	1,2,3,4	2
K2	1,2,3,4	2,3,4
Q1	1,2,3,4	Ø
Q2	1,2,3,4	1,2,4

Let us run the algorithm in its entirety with any and all other possible assignment of variables and see if we can find a solution.

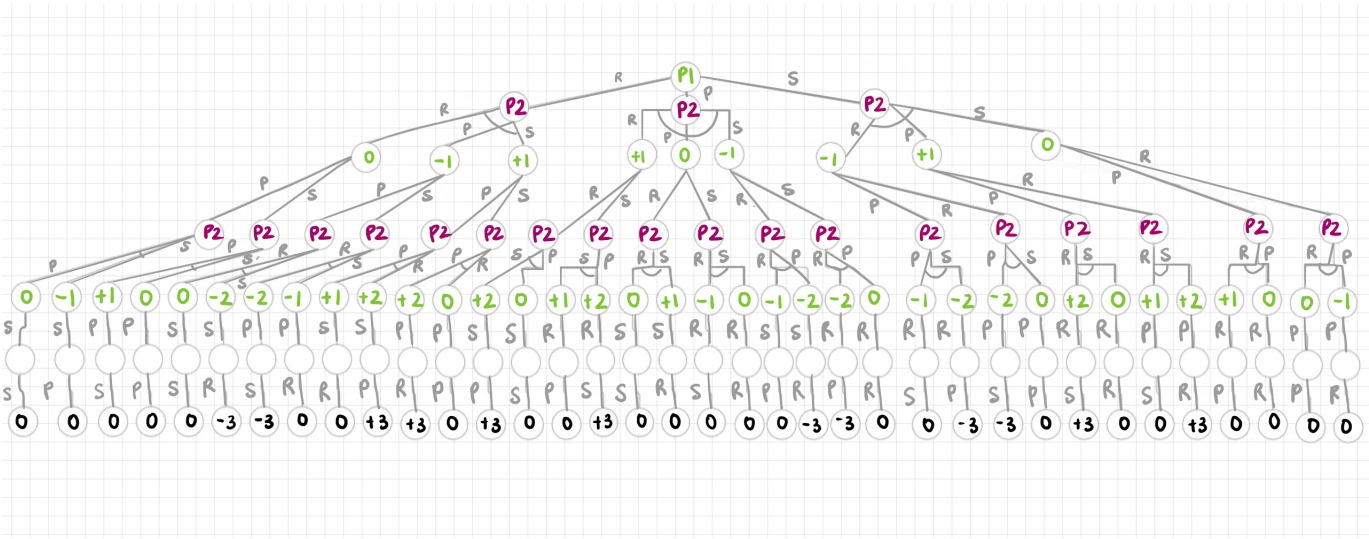
	Nothing Assigned	K1 = 1	K2 = 1	K2 = 2	K2 = 4	K1 = 2	K1 = 3	K1 = 4	k2=1	K2=3	K2=4
K1	1,2,3,4	1	1	1	1	2	3	4	4	4	4
K2	1,2,3,4	1,2,3	1	2	4	1,2,3	2,3,4	1,3,4	1	3	4
Q1	1,2,3,4	4	4	Ø	Ø	Ø	Ø	1	Ø	Ø	1
Q2	1,2,3,4	2,3	Ø	Ø	Ø	1,3,4	2,4	2,3	Ø	Ø	Ø

2 Question 2: Search and Game Playing

You are playing rock-paper-scissors with a friend but with a twist. The game consists of three rounds and each round, players may not play something that they have played before. So if you play rock the first round, you cannot play it again in rounds two and three. The winner of the game is the player who wins the most rounds.

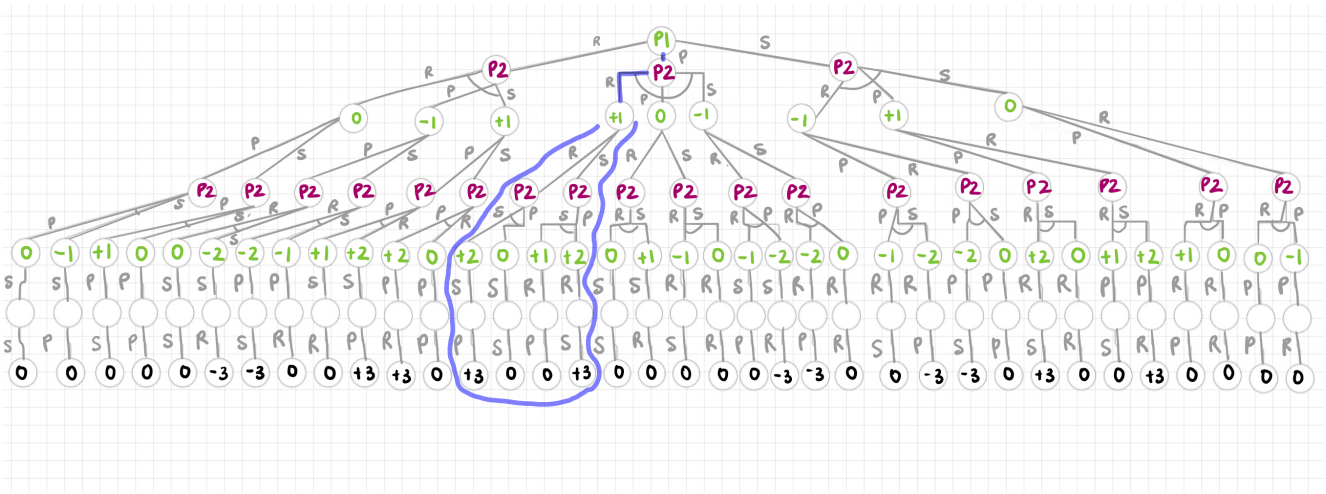
a) Draw the AND-OR tree associated with this game

Recall that OR nodes are when branching is induced by an agent's choice between actions (meaning we choose what action we want to play) and AND nodes are when branching is induced by the environments choice of outcome (meaning that we branch when the related action is stochastic - the other opponents (non-predictable) actions.



Since the description of the game does not specify how to handle ties among either the mini matches or the overall game, I specifies (showed in graph) that the utility of each mini match is +1,0,-1. As the games progress, the utility becomes the sum of the past mini matches played in that specific branch. We observe at the end that there are many ties, this occurs either because the two players tie within one minimatch and win/lose another mini match respectively or because they tied all mini matches. The graph begins with player 1 (us) picking R,P,S and then immediately followed by AND branch of the possible choices player 2 can make. Because we cannot repeat out choices, our branching factor (and that of the opponent) decreases by one at each mini match. The final row depicts the sum of utilities at the end of each game, there are only 1/3 chances of a round of minimatches not ending in a tie of any kind with this approach.

b) In the first round, you play paper and your friend plays rock, giving you the first win. Can you guarantee a win of the game overall? Explain by extracting a contingency plan from the AND-OR tree above. Check your answer by reformulating the tree into a game tree where you apply mini max algorithm.

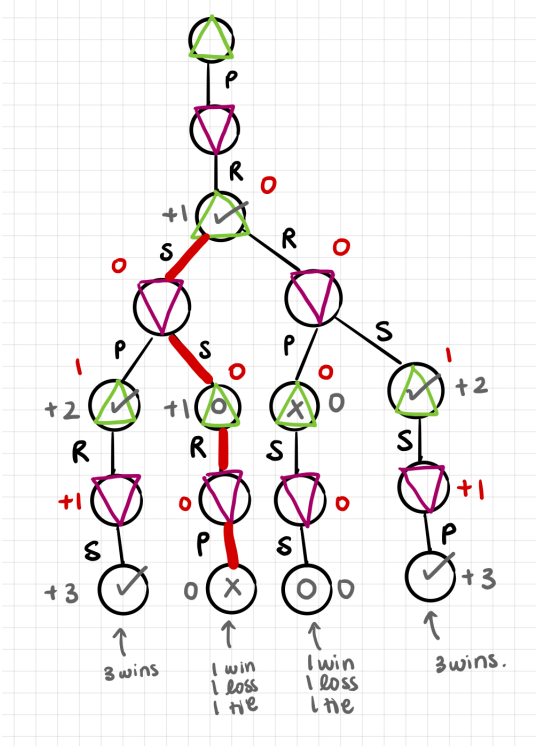


Looking at the subtree given to us by the first two choices of the players, we can see that we cannot necessarily guarantee a win but rather we can guarantee that P2 will not win. Given the choices that we ruled out with the first mini-match we see that there is no outcome for which P2 ends with 2 won minimatches leading to a overall win. Of the four potential outcomes, 2 lead to victory for P1 and 2 lead to ties.

Contingency plan: starting from round 2 we can do the following:

- Play Rock. If P2 plays Paper, then our last move (scissors) will result in a tie. If P2 plays Scissors, then our last move will lead to a win.

If we reform out tree into a game tree we have the following. Let us apply the minimax algorithm.



We can see that running the mini max algorithm gives us a tie because P2 wants to minimize the worst case utility that it has been given after the first round and similarly P1 can only try to maximise the worst case utility that it gets as its options begin to be limited as they cannot repeat a move.

3 Question 3: Propositional Logic

a) How many models are there for each of the following statements?

1. $(A \wedge B) \vee C$

   We know that a model is an assignment of truth values, one for every propositional symbol. Let the following table show the possible models for the statement above:

m_i	A	B	C	Result
m_1	T	T	T	T
m_2	T	T	F	T
m_3	T	F	F	F
m_4	F	F	F	F
m_5	F	F	T	T
m_6	F	T	T	T
m_7	T	F	T	T
m_8	F	T	F	F

   It is clear then that the number of models that we achieve for this statement is 2^3 where the exponent is the # of variables in the statement.

2. $A \vee (A \Rightarrow B) \wedge \neg B$

m_i	A	B	Result
m_1	T	T	F
m_2	T	F	T
m_3	F	T	F
m_4	F	F	T

3. $\neg(A \wedge B \wedge C \wedge D \wedge E) \vee (B \wedge C)$ In order to save time and space, the table of this statement has been omitted but we know that there are 2^5 models for this statement. Following the logic presented in the first question.

b)State whether each of the following is valid, unsatisfiable or satisfiable. Support you r answer with a truth table or proof

- $((A \Rightarrow B) \wedge (A \vee C)) \Rightarrow (B \vee C)$

m_i	A	B	C	A $\Rightarrow$ B	(A $\Rightarrow$ B) $\wedge$ (A $\vee$ C)	(B $\vee$ C)	Result	
m_1	T	T	T	T	T	T	T	T
m_2	T	T	F	T	T	T	T	T
m_3	T	F	T	F	T	F	F	F
m_4	T	F	F	F	T	F	F	T
m_5	F	T	T	T	T	T	T	T
m_6	F	T	F	T	F	F	T	T
m_7	F	F	T	T	T	T	T	T
m_8	F	F	F	T	F	F	F	T

Thus valid.

- ((A $\vee$ B) $\Rightarrow$ C) $\Rightarrow$ ((A $\Rightarrow$ C) $\wedge$ (B $\Rightarrow$ C))

m_i	A	B	C	(A $\vee$ B)	(A $\vee$ B) $\Rightarrow$ C	(A $\Rightarrow$ C)	(B $\Rightarrow$ C)	((A $\Rightarrow$ C) $\wedge$ (B $\Rightarrow$ C)	Result
m_1	T	T	T	T	T	T	T	T	T
m_2	T	T	F	T	F	F	F	F	T
m_3	T	F	T	T	T	T	T	T	T
m_4	T	F	F	T	F	F	T	F	T
m_5	F	T	T	T	T	T	T	T	T
m_6	F	T	F	T	F	T	F	F	T
m_7	F	F	T	F	T	T	T	T	T
m_8	F	F	F	F	F	T	T	T	T

Thus valid.