Project Summary

- The game Minesweeper consists of a board with some tiles containing bombs and other tiles containing numbers corresponding to the number of bombs in its perimeter
- Our project aims to determine the next plausible move in a game of Minesweeper
- The program is supplied with a starting board with some tiles already revealed. All unrevealed tiles are initially labelled as 'unknown'
- The program will analyze this given board:
    - The program loops over every tile on the board and checks each tile's perimeter. This analysis is performed in order to determine the status of as many 'unknown' tiles on the board as possible
    - A tile will be labelled 'safe' if it is confirmed a bomb is not on the tile
    - A tile will be labelled 'unsafe' if it is confirmed a bomb is on the tile
    - A tile will remain 'unknown' if it cannot be satisfied by any of the models

Propositions

The sub-variables (i, j) will be attached to each proposition and indicate the coordinates of the tile. (i, j) = (0, 0) will indicate the top left position of the board. The sub variable 'a' will be attached to revealed tiles and indicate the value of said tile (1, 2, 3, or 4).

B(i, j) indicates there is a bomb at position i,j.

~B(i,j) indicates there is no bomb at position i,j.

S(i, j) indicates the tile at i,j is safe.

P(i,j) indicates the tile at position i,j is unknown or cannot be confirmed.

Q(a, i, j) indicates there is a numbered tile of value *a* at position i, j.

Constraints

A square cannot have both a bomb and a numbered tile on it:

¬(B(i,j) ∧ Q(a,i,j))

An unknown square that does not have a bomb on it will be labelled as safe.

(U(i,j) ∧ ¬B(i,j)) → S(i,j)

A numbered square Q(a,i,j) must have the same number of Bombs (B(i,j)) surrounding it as the value of a. This means that, for example, an unknown tile next to Q(1,i,j) can be determined to not be a bomb if there is already a known bomb on a square next to Q as there cannot be more than one bomb around Q.

e.g.: (Q(1,0,0) ∧ B(0,1) ∧ U(1,0)) → S(1,0)

This also means that the number of bombs around Q cannot be less than a, so any number of unknown squares can be determined to be bombs if there are the same number of unrevealed squares as the value of A. For example:

(Q(2,0,0) ∧ Q(1,0,1) ∧ U(1,0) ∧ U(1,1)) → (B(1,0) ∧ B(1,1))

Both of the above examples can be written generally for a constraint centered around the square Q(a,i,j). For this constraint, let x and y represent a square where {i-1 < x < i+1} and {j-1 < y < j+1} and (x,y) ≠ (i,j). Each (x,y) combination will have a superscript t where each (x,y)^t represents a different square.

For a=1,

This first constraint represents a square Q where it is touching one bomb on any of the squares around it, and states that all of the other surrounding squares (it is implied that these squares exist) must all be safe.

(Q(1,i,j) ∧ B(x,y)^1) → (S(x,y)^2 ∧ S(x,y)^3 ∧ S(x,y)^4 ∧ S(x,y)^5 ∧ S(x,y)^6 ∧ S(x,y)^7 ∧ S(x,y)^8)

For the below constraint, the unknown a value for Q(a,i,j) do not matter. The only one that matters is the known Q(1,i,j). The constraint below represents a square Q which is touching one unknown square and the rest of the squares around it are revealed to be safe. From this, we can confirm that the unknown square must be a bomb.

(Q(1,i,j) ∧ U(x,y)^1 ∧ (Q(a,x,y)^2 ∨ S(x,y)^2) ∧ (Q(a,x,y)^3 ∨ S(x,y)^3) ∧ (Q(a,x,y)^4 ∨ S(x,y)^4) ∧ (Q(a,x,y)^5 ∨ S(x,y)^5) ∧ (Q(a,x,y)^6 ∨ S(x,y)^6) ∧ (Q(a,x,y)^7 ∨ S(x,y)^7) ∧ (Q(a,x,y)^8 ∨ S(x,y)^8)) → B(x,y)^1

For a=2,

$(Q(2,i,j) \land B(x,y)^1 \land B(x,y)^2) \rightarrow (S(x,y)^3 \land S(x,y)^4 \land S(x,y)^5 \land S(x,y)^6 \land S(x,y)^7 \land S(x,y)^8)$

$(Q(2,i,j) \land U(x,y)^1 \land U(x,y)^2 \land (Q(a,x,y)^3 \lor S(x,y)^3) \land (Q(a,x,y)^4 \lor S(x,y)^4) \land (Q(a,x,y)^5 \lor S(x,y)^5) \land (Q(a,x,y)^6 \lor S(x,y)^6) \land (Q(a,x,y)^7 \lor S(x,y)^7) \land (Q(a,x,y)^8 \lor S(x,y)^8)) \rightarrow (B(x,y)^1 \land B(x,y)^2)$

For a=3,

$(Q(3,i,j) \land B(x,y)^1 \land B(x,y)^2 \land B(x,y)^3) \rightarrow (S(x,y)^4 \land S(x,y)^5 \land S(x,y)^6 \land S(x,y)^7 \land S(x,y)^8)$

$(Q(3,i,j) \land U(x,y)^1 \land U(x,y)^2 \land U(x,y)^3 \land (Q(a,x,y)^4 \lor S(x,y)^4) \land (Q(a,x,y)^5 \lor S(x,y)^5) \land (Q(a,x,y)^6 \lor S(x,y)^6) \land (Q(a,x,y)^7 \lor S(x,y)^7) \land (Q(a,x,y)^8 \lor S(x,y)^8)) \rightarrow (B(x,y)^1 \land B(x,y)^2 \land B(x,y)^3)$

For a=4,

$(Q(4,i,j) \land B(x,y)^1 \land B(x,y)^2 \land B(x,y)^3 \land B(x,y)^4) \rightarrow (S(x,y)^5 \land S(x,y)^6 \land S(x,y)^7 \land S(x,y)^8)$

$(Q(4,i,j) \land U(x,y)^1 \land U(x,y)^2 \land U(x,y)^3 \land U(x,y)^4 \land (Q(a,x,y)^5 \lor S(x,y)^5) \land (Q(a,x,y)^6 \lor S(x,y)^6) \land (Q(a,x,y)^7 \lor S(x,y)^7) \land (Q(a,x,y)^8 \lor S(x,y)^8)) \rightarrow (B(x,y)^1 \land B(x,y)^2 \land B(x,y)^3 \land B(x,y)^4)$

Model Exploration

There are various ways the logic for this problem could have been laid out. The project started out with the goal of being able to lay out the logic necessary to solve a whole game of minesweeper on a sample 3x3 board. The plan was to determine if a tile is safe and reveal its value. After some work and utilizing the feedback given, we have decided to create a logical model to base if whether any of the spots on the board are safe and not reveal their values, to compute 1 step of the game.

The feedback put into perspective how the complexity of the game and given variables will substantially increase if the whole board was to be solved (Q(a,i,j), a will increase all the way up to 8 on large scale boards). Creating a logical model for 1 step will allow us to model the logic on a smaller scale board and later expand to larger more complicated models if needed.

Our python model will look over all the tiles individually with the layout of the given scenario (being a constraint) and store the values of the map into a 2D array (each sub array storing a row). The perimeter of all the tiles are analyzed one at a time, in order to decide if the tiles are either a bomb Q(i,j), safe S(i,j), unknown U(i,j)

Jape Proof Ideas

1.) $U \vdash (B \lor Q \lor S) \land ((\neg B \lor \neg S) \land (\neg Q \lor \neg S) \land (\neg B \lor \neg Q))$. An unknown spot is an exclusive or with all possibilities but only one of the possibilities at a time

2.) $(B \lor S \lor Q), \neg S, Q \rightarrow S \vdash B$. Given that a tile must be either a bomb, safe, or numbered, if we know the tile is not safe and that if it is numbered it must be safe, we can deduce that the tile is a bomb.

3.) $T \rightarrow Q2, T, Q1, Q2 \rightarrow (\neg B \land \neg S \land \neg P), Q1 \rightarrow (B \lor Q2 \lor S \lor P) \land (\neg B \lor \neg Q2 \lor \neg S) \land (\neg B \lor \neg Q2 \lor \neg P) \land (\neg B \lor \neg P \lor \neg S) \vdash (Q2 \land \neg B \land \neg S \land \neg P)$. If there exists a revealed tile (Q1), we know that to its left is exclusively safe (S) or numbered (Q2) or bomb (B) or unknown (P). Then, if elsewhere in the initial board layout (T) there exists a numbered tile in that spot (to the left of Q1), that tile will be Q2. Then, when Q1 sees this Q2, it will know that the tile to its left is exclusively Q2. In other words, this is meant to be a way for a tile to determine what types of tiles exist in its perimeter.

Requested Feedback

Questions:

1. Can we get some idea as two how the jape proof structuring would be implemented. We have made some Jape proofs based off our understanding of the logic, but could use some feedback to make more sense of the grey areas. We now have a much better general idea on how to make this logic work, but any to take the extra step and fully flesh out our understanding we need some pointers.

2. We are having trouble generalizing our constraints in propositional logic because of the positional variables in each instance. We are thinking that we may have to write out a constraint for every possible scenario, which we want to avoid. What advice could you give on how to generalize our constraints?

3. We are confused on how to begin with the python implementation. Could we get some advice on how we should approach implementing this logic in a python model?