Group 17: Andrew Adams, Cameron De Matos, Ash Taraghi, Sam Vousden

Project Summary

- The game Minesweeper consists of a board with tiles. These tiles can either be a bomb or a numbered tile. The number of a tile refers to the number of bombs in the perimeter of that tile.
- Our project aims to determine the next logical move in a game of Minesweeper based on the rules of the game.
- 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 its 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) are used to denote (row, column) and will be attached to each proposition. For example, $(i, j) = (0, 0)$ will indicate the top left tile of the board. The sub-variable a will be attached to revealed tiles and indicates the number on said tile (e.g.: 1, 2, 3, or 4).

$B_{(i,j)}$ indicates there is a bomb at position (i, j).

$\neg B_{(i,j)}$ indicates there is no bomb at position (i, j).

$S_{(i,j)}$ indicates the tile at (i, j) is safe.

$U_{(i,j)}$ 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 tile cannot have both a bomb and a numbered tile on it: $\neg(B_{(i,j)} \wedge Q_{(a,i,j)})$
- A tile cannot simultaneously be a bomb and be safe: $\neg(B_{(i,j)} \wedge S_{(i,j)})$
- If a tile is not a bomb, then it must be safe. If a tile is safe, then it must be numbered: $\neg B_{(i,j)} \rightarrow (S_{(i,j)} \rightarrow Q_{(a,i,j)})$
- If a tile is not safe, then it must be a bomb: $\neg S_{(i,j)} \rightarrow B_{(i,j)}$
- A numbered tile $Q_{(a,i,j)}$ must have the same number of bombs ($B_{(i,j)}$) surrounding it as the value of *a*. For example, if an unknown tile is in the perimeter of $Q_{(1,i,j)}$ can be determined to be safe if there is already 1 known bomb on a tile in the perimeter of $Q_{(1,i,j)}$. For example:

  $(Q_{(1,0,0)} \wedge B_{(0,1)} \wedge U_{(1,0)}) \rightarrow S_{(1,0)}$

  This also means that the number of bombs in the perimeter of $Q_{(a,i,j)}$ cannot be less than *a*. Therefore, any number of unknown tiles in the perimeter of $Q_{(a,i,j)}$ can be determined to be bombs if there are the same number of unknown tiles as the value of *a*. In other words, if there are *n* unknown tiles in the perimeter of $Q_{(n,i,j)}$, then we can deduce that each of those *n* tiles is a bomb. For example:

  $(Q_{(2,0,0)} \wedge S_{(0,1)} \wedge U_{(1,0)} \wedge U_{(1,1)}) \rightarrow (B_{(1,0)} \wedge B_{(1,1)})$

  Both of the above examples can be written in a general way for a constraint centered around the square $Q_{(a,i,j)}$. For this constraint, let *x* and *y* represent a tile where $\{i\text{-}1 < x < i\text{+}1\}$ and $\{j\text{-}1 < y < j\text{+}1\}$ respectively, and $(x, y) \neq (i, j)$. Each (x, y) combination will have a superscript *t* where each $(x, y)^t$ represents a different tile in the perimeter of $Q_{(a,i,j)}$.

  As an example, let $a = 1$:

  This first constraint represents a tile $Q_{(1,i,j)}$ where there is one bomb in its perimeter, and states that all the other surrounding tiles – it is assumed such tiles exist - must all be safe:

  $(Q_{(1,i,j)} \wedge B_{(x,y)\text{^}1}) \rightarrow (S_{(x,y)\text{^}2} \wedge S_{(x,y)\text{^}3} \wedge S_{(x,y)\text{^}4} \wedge S_{(x,y)\text{^}5} \wedge S_{(x,y)\text{^}6} \wedge S_{(x,y)\text{^}7} \wedge S_{(x,y)\text{^}8})$

For the constraints below, the unknown *a* value in $Q_{(a, i, j)}$ does not matter. The only one that matters is the known $Q_{(1, i, j)}$. The constraints below represent a tile Q which is touching *a* unknown tiles and the rest of the tiles in its perimeter are revealed to be safe. From this, we can confirm that the unknown tile(s) must be a bomb/bombs:

For *a* = 1:

$(Q_{(1,i,j)} \land U_{(x, y)^1} \land S_{(x, y)^2} \land S_{(x, y)^3} \land S_{(x, y)^4} \land S_{(x, y)^5} \land S_{(x, y)^6} \land S_{(x, y)^7} \land S_{(x, y)^8}) \longrightarrow B_{(x, y)^1}$

For *a* = 2,

$(Q_{(2, i, j)} \land B_{(x, y)^1} \land B_{(x, y)^2}) \longrightarrow (S_{(x, y)^3} \land S_{(x, y)^4} \land S_{(x, y)^5} \land S_{(x, y)^6} \land S_{(x, y)^7} \land S_{(x, y)^8})$

$(Q_{(2, i, j)} \land U_{(x, y)^1} \land U_{(x, y)^2} \land S_{(x, y)^3} \land S_{(x, y)^4} \land S_{(x, y)^5} \land S_{(x, y)^6} \land S_{(x, y)^7} \land S_{(x, y)^8}) \longrightarrow (B_{(x, y)^1} \land B_{(x, y)^2})$

For *a* = 3,

$(Q_{(3, i, j)} \land B_{(x, y)^1} \land B_{(x, y)^2} \land B_{(x, y)^3)} \longrightarrow (S_{(x, y)^4} \land S_{(x, y)^5} \land S_{(x, y)^6} \land S_{(x, y)^7} \land S_{(x, y)^8})$

$(Q_{(3, i, j)} \land U_{(x, y)^1} \land U_{(x, y)^2} \land U_{(x, y)^3} \land S_{(x, y)^4} \land S_{(x, y)^5} \land S_{(x, y)^6} \land S_{(x, y)^7} \land S_{(x, y)^8}) \longrightarrow (B_{(x, y)^1} \land B_{(x, y)^2} \land B_{(x, y)^3})$

For *a* = 4,

$(Q_{(4, i, j)} \land B_{(x, y)^1} \land B_{(x, y)^2} \land B_{(x, y)^3} \land B_{(x, y)^4}) \longrightarrow (S_{(x, y)^5} \land S_{(x, y)^6} \land S_{(x, y)^7} \land S_{(x, y)^8})$

$(Q_{(4, i, j)} \land U_{(x, y)^1} \land U_{(x, y)^2} \land U_{(x, y)^3} \land U_{(x, y)^4} \land S_{(x, y)^5} \land S_{(x, y)^6} \land S_{(x, y)^7} \land S_{(x, y)^8}) \longrightarrow (B_{(x, y)^1} \land B_{(x, y)^2} \land B_{(x, y)^3} \land B_{(x, y)^4})$

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 increase substantially 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 perimeters of all the tiles are analyzed one at a time to decide if the tiles are either a bomb $Q_{(i, j)}$, safe $S_{(i, j)}$, unknown $U_{(i, j)}$.

Jape Proofs

1.) ¬B →S, Q → S, ¬S ⊢ B ∧¬Q If a tile is not a bomb, we know that it is safe. If a tile is a numbered tile this also implies that it is safe. If we have a not safe tile, this means that we know it a bomb and not a numbered tile.

1:	¬B→S, Q→S, ¬S	premises
2:	¬S→¬Q	Theorem P→Q ⊢ ¬Q→¬I 1.2
3:	¬Q	→ elim 2,1.3
4:	¬S→¬¬B	Theorem P→Q ⊢ ¬Q→¬I 1.1
5:	¬¬B	→ elim 4,1.3
6:	B	Theorem ¬¬P ⊢ P 5
7:	B∧¬Q	∧ intro 6,3

2.) (B ∨ S ∨ Q), ¬S, Q → S ⊢ 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.

1:	(B∨S∨Q), ¬S, Q→S	premises
2:	B∨S	assumption
3:	B	assumption
4:	S	assumption
5:	⊥	¬ elim 4,1.2
6:	B	contra (constructive) 5
7:	B	∨ elim 2,3-3,4-6
8:	Q	assumption
9:	S	→ elim 1.3,8
10:	⊥	¬ elim 9,1.2
11:	B	contra (constructive) 10
12:	B	∨ elim 1.1,2-7,8-11

3.) T ⟶ Q2, T, Q1, Q2 ⟶ (¬B ∧ ¬S ∧ ¬P), Q1 ⟶ (B ∨ Q2 ∨ S ∨ P) ∧ (¬B ∨ ¬Q2 ∨ ¬S) ∧ (¬B ∨ ¬Q2 ∨ ¬P) ∧ (¬B ∨ ¬P ∨ ¬S) ⊢ (Q2 ∧ ¬B ∧ ¬S ∧ ¬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.

1: T→Q2, T, Q1, Q2→(¬B∧¬S∧¬P), Q1→(B∨Q2∨S∨P)∧(¬B∨¬Q2∨¬S)∧(¬B∨¬Q2∨¬P)∧(¬B∨¬P∨¬S)	premises
2: Q2	→ elim 1.1,1.2
3: ¬B∧¬S∧¬P	→ elim 1.4,2
4: ¬B∧¬S	∧ elim 3
5: ¬S	∧ elim 4
6: ¬B	∧ elim 4
7: Q2∧¬B	∧ intro 2,6
8: Q2∧¬B∧¬S	∧ intro 7,5
9: ¬P	∧ elim 3
10: Q2∧¬B∧¬S∧¬P	∧ intro 8,9

First Order Extension

Modeling with Predicate Logic:

Constructing Predicates

Variables: *a* (indicates the number of bombs adjacent to a square), *i* (indicates the row number), *j* (indicates column number). Each variable is bound to a value in its respective domain to create an instantiation of the predicate.

Domain: The domain of *i* and *j* is the set of row/column positions on the board. For the sake of modeling with simplicity, suppose the size of the board for this demonstration is 3. This means the domain of *i* and *j* is the set of {1, 2, 3} and *i* and *j* can be bound to any of these numbers.

The domain of a is the set of the number of bombs beside a tile. Again, for the sake of simplicity, suppose the maximum number of bombs that can be beside a tile at any given moment is 2. The domain of a is thus {0, 1, 2}.

Predicates:

B(*i, j*) “There is a mine at row *i* column *j*.”

Q(*a, i, j*) “There are mines in *a* of the squares adjacent to the square located at row *i*, column *j*.”

Universal Quantification

We assert the predicate must hold for all values of the variable:

$\forall i.\forall j.$B(*i, j*) “All row-column positions on the board contains a bomb”

This is the equivalent to the conjunction B(1, 1) ∧ B(1, 2) ∧ B(1, 3) ∧ B(2, 1) ∧ B(2, 2) ∧ B(2, 3) ∧ B(3, 1) ∧ B(3, 2) ∧ B(3, 3). This statement will always evaluate to false: there will never be a bomb on every tile of the board

Existential Quantification

We assert that there exists a set of values for the variables such that the predicate will hold true:

$\exists i.\exists j.$B(*i, j*) “A row-column position of the board contains a bomb”

This is equivalent to the disjunction B(1, 1) ∨ B(1, 2) ∨ B(1, 3) ∨ B(2, 1) ∨ B(2, 2) ∨ B(2, 3) ∨ B(3, 1) ∨ B(3, 2) ∨ B(3, 3).

In this case, the statement will always evaluate to true as there must always be at least one bomb on the board

Constraints

Using this knowledge, we can create some simple and complex constraints. Suppose we know there is only one bomb on the entire 3x3 board. We can create the constraint:

$$\forall i.\forall j.\forall i'.\forall j'.(\text{B}(i, j) \land (i \neq i' \lor j \neq j')) \rightarrow \neg\text{B}(i' , j')$$

This reads as “For every *i*, *j*, *i’*, *j’* in their respective domains if there is a bomb at row-column *i*, *j* and either *i’* is different from *i* or *j’* is different from *j*, then position *i’*, *j’* does not contain a bomb.

A more complex constraint would look like:

$$\forall i1.\forall j1.\forall i'.\forall j'.(\text{B}(i1, j1) \land (|i1 - i'| \leq 1) \land (|j1 - j'| \leq 1) \land \neg(i1 = i' \land j1 = j') \land (\forall i2.\forall j2.((|i2 - i'| \leq 1) \land (|j2 - j'| \leq 1) \land (i1 \text{ != } i2 \lor j1 \text{ != } j2)) \rightarrow \text{B}(i2, j2))) \rightarrow \text{A}(i' , j', 1)$$