# Project Summary

This project aims to generate a list of moves to get from an initial board to the solved board where all number tiles are in the correct positions.

A model will correspond to a 3x3 board with numbered tiles randomly placed in 8 of the 9 positions. The model produces a list of moves that solves the board.



# Propositions

* Atij: True when tile t is in the position (i,j)
  + This proposition checks whether a tile is in a given position.
* Xij: True when position (i,j) corresponds to the correct tile number.
* W: True when all Xij are true.
  + When true, the board is in its goal state
* Bij: True when position (i,j) corresponds to no number tiles (position is blank)
* Uij: True when Ti-1j is true (Up\above)
  + When true, the position above the given position is a valid position on the board.
* Dij: True when Ti+1j is true (Down\below)
  + When true, the position below the given position is a valid position on the board.
* Lij: True when Tij-1 is true (Left)
  + When true, the position to the left of a given position is a valid position on the board.
* Rij: True when Tij+1 is true (Right)
  + When true, the position to the right of a given position is a valid position on the board.
* Tpq: True if p is 0,1, or 2, and q is 0, 1, or 2. (if p,q is on the board)
  + When true, the given position is a valid position on the board.
* Sijpq: True if a tile can move from position (i,j) into position (p,q)

# Constraints

* The initial tile has to be a blank in order to swap with a target tile
  + Sijpq → Bij
* A blank tile can only swap with a position above, below, or beside it that is on the board
  + Sijpq → Uij | Dij | Lij | Rij
* Constraints to check if the tile above, below, left, or right is a valid position
  + U ij → Ti+1j
  + Dij → Ti-1j
  + Rij → Tij+1
  + Lij → Tij-1
* The board can only be in its goal state when W is true, such that
  + W → X00 & X01 &…&X22
* X00 is true when tile 1 is assigned to position (0,0), X01 is true when tile 2 is assigned to position (0,1)... etc for all tiles.
  + X00 >> A1,00
  + X01 >> A2,01
  + X02 >> A3,02
  + X10 >> A4,10
  + X11 >> A5,11
  + X12 >> A6,12
  + X20 >> A7,20
  + X21 >> A8,21
  + X22 >> A’blank’,22

# Model Exploration

* We set up our model so that, given an initial board, the model should give us a list of moves that solves the board
  + The initial board would let the model assign tiles to positions initially, and then the assignments change with each move (using our Assigned propositions)
* We changed how we looked at the problem: at first we were viewing “moves” as tiles swapping with each other to get to the right positions, however we simplified the setting by viewing it as the “blank” tile swapping with the neighbouring number tiles repeatedly.
  + That led us to reformat our constraints so that we set Bij as always true, and thus, we do not need to check whether the tile swapping with the initial tile is blank, since the initial tile will always be the ‘blank’ tile. Specifically, when assessing the above, below, left, and right propositions, we did not need to add the constraint that Bpq had to be true for the direction propositions (Uij , Dij , Lij , Rij) to be true.
  + This saves us a lot of computing time because the model will not have to check each tile on the board to see which ones were neighbouring a blank and thus could move into the spot. Now the model only needs to check the propositions for the ‘blank’ tile on the board for each swap.

# Jape Proof Ideas

* Two of our constraints can be combined to determine whether a swap can happen or not, such that: (Bij & (Aij | Dij | Lij | Rij ) )→ Sijpq, and we can prove the last sequent through the following steps:
  + - Initial premise: (Bij & (Aij | Dij | Lij | Rij ) ) → Sijpq,
    - ~ (Bij & (Aij | Dij | Lij | Rij ) ) | Sijpq
    - ~Bij | ~(Aij | Dij | Lij | Rij ) ) | Sijpq
    - conclusion: ~Bij | ~Aij & ~Dij & ~Lij & ~Rij  | Sijpq

This used a few of the proof strategies learnt in class such as de Morgans Law and implication removal. We plan to make a few more for our final submission.

# Requested Feedback

* We have a bunch of constraints, but many of them are quite short with only one implication. We were hoping for feedback for how we could come up with more good quality jape proofs for our project.
* We would like some feedback on how we implemented the propositions and constraints that check whether the position above/below/left/right the current position exists and is a valid position on the board.

# First-Order Extension

*Describe how you might extend your model to a predicate logic setting, including how both the propositions and constraints would be updated.* ***There is no need to implement this extension!***