# Project Summary

Laser chess is a game like chess. It is a game with two sides where both teams have a laser, triangular pieces that have one mirrored side and a king piece with no mirrors. The aim of the game is to use the mirrored pieces on the board to bounce the laser off the mirrors so to laser can reach the king and hit him. Once a team hits the opponents king with their laser they win. During your move you can either move a piece or you king one position in any directions or rotate a piece 90 degrees. The laser always starts at the same position. At the end of your turn, you use your laser and see where it ends. If it goes off the map nothing happens and it is your opponent’s turn, if it hits the non-mirrored side of a piece no matter what team that piece is on that piece is removed from the board and if it hits a king the game is over and the team with the king left standing wins. For our modeling projecting we will be checking if it is possible for the laser to reach the king with our laser in one move on a 5x4 board. 4 mirrored pieces that will all be on our team will be randomly placed on the board to help direct the laser to the king.

# Propositions

P(x,y),(o) , every piece has an (x, y) position, (o) orientation.

* Ex: P(1,1) (NW) would be True for a piece in the position (1,1) with an orientation of NW
* o can be northwest (NW), southwest (SW), southeast (SE) and northeast (NE), which describes the side with the mirror

L(x,y), (d), laser has an (x, y) position and (d) for the direction of the laser.

* Ex: L(2,3) (N) would be True if the laser is at the position (2,3) going towards N.
* d direction can be north (N), south (S), east (E), west (W)

K(x,y)  opponents king that has a position (x, y).

G is the proposition for game over.

* Ex: G is True if the opponent's king is hit by the laser

# Constraints

G only holds when the laser and the king have the same position.

* (K2,2 ∧ L2,2, N) G

A piece can’t move to a position where there is another piece.

The laser always starts at the same position with the same orientation.

* L(0,0) (E)

The king always starts on the same x and y coordinates.

* K(4, 3)

The laser keeps going in the same direction until it hits a mirror side of piece, goes out of bounds, hits the king, or hits a non-mirror side.

* Ex: L(2,3) (E) ∧ P(2,3)(NW)  L(2,3)(N)

A piece can only move to one adjacent square or rotate 90 degrees.

The 4 pieces must be on the board

Only one piece can be at a certain position at a time

# Model Exploration

In our model, we made a 5x4 board by giving each piece and King their own grid of Booleans. The board returns true if there is a piece at that position. Our goal was to try to make code that checked if there was a way to win from a series of test cases, by moving one of the pieces. We wrote code for a handful of classes, 1 for pieces, laser and king components. These all held the arrays of true or false statements for each piece, laser or king. This was important because it was an easy way to represent the position of each component, and we could manipulate them with functions we made for each class (move up, down, etc). Then we added all the constraints to play as the rules of the game, and had more functions that worked to check every spot around each piece, and whether winning the game was possible after just moving or rotating one piece at most one position.

Upon further exploration, the game created more and more issues, meaning we had to limit our variables more and more. Originally, we wanted to do a game against and ai, which seemed extremely difficult to implement in this way. We also had to lock the king position, because it was getting more and more tedious to write out every move that was possible.

# Jape Proofs

First:

Graphical user interface, application, table

Description automatically generated

In this first proof, the propositions represent different spots in one piece’s domain (it’s grid). P1 would be one x and y coordinate, and P2 would be another. They both can’t be true, but one or the other can and they can both be false. It’s a very simple concept but important for the basis of how the pieces work in the game, because one piece can’t be at multiple spots on the board.

Second:

Table

Description automatically generated

In the second proof, P is true if a piece is at a certain spot on the board, E is true if that spot is empty, Q is true if the king is at that spot, A is true if the laser is at the current spot, B is true if the game is won, D is true if the laser is changing direction at the current position. The propositions are; there must be either a piece, a king, or the spot is empty, there is a laser at the spot, if the king and the laser are at the spot the game ends, the direction does not change, if there’s a piece at the spot there is a direction change, the spot is not empty. This is important because it is the win condition: it’s trying to prove a game over.

Third:

Text

Description automatically generated

A is true if there is a laser at a specific spot on the grid, T is true if this spot is actually the edge of the board(hits the edge as if to go beyond, after any obstacles on the actual board), Q is true if the king is at the position, C is true if the laser will continue after this position and B is true if the game is won. The propositions are as follows: there is a laser at this spot, it is hitting the edge of the board, if this is the edge, then it can’t be a king piece (because it’s technically not on the board), if there’s a laser hitting the edge, then the laser must stop, if the laser must stop and it’s not hitting a king this means there is no win. All of these propositions prove that there is a loss.

# First Order Extension

One thing we can use in predicate logic is coordinates, and it make a lot more sense where it’s a grid of true and false values.

For all pieces, they should have a x and y and a direction, E.g. P(x,y,d), true where that piece is and what direction it’s facing

The king should have only and x and y, K(x,y) representing where the piece is when it’s true

The laser, similar to the piece, has a position and a direction L(x,y,d)

All of the x, y and d values have different domains, and the values in the propositions must stay within these domains.

Game ends at if L(x,y,d) and K(x,y) is true (share same x and y values) or if the laser is going in the direction of the border as it’s in the outside ring of the grid(facing the edge)

For example, trying to express that a piece can’t be at any other spot can be expressed as xy.i.zr.j.((P(x,y,i) P(z,r,j)), but this is only true for accessing DIFFERENT positions, where (x,y,i) does not = (z,r,j). There isn’t really a way to write the exception in logic. For example, if there’s a piece P(1,2,3), then that piece can’t exist at P(3,1,2), but of course there is a piece at P(1,2,3) because we just stated there was.

Another extension that could be done would be writing that there always exists a piece position: . Each piece has a spot that it is true at, and as we saw in the last example no more than one.

Explaining that there is either a piece, a king, or a spot is empty can be shown by: xy.((P(x,y) ) ), E representing a set of empty spots represented by true when it’s empty. This is logically equivalent to an exclusive or, where only one can be true for each x and y value.