# **Confirming Hexapawn**

Group 49

Yutong Jiang

Kevin Tran

Colin Wasik

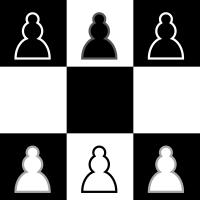
Andrew Wilker

# Project Summary

For our project, we are confirming Hexapawn.

Hexapawn is a game played on a 3x3 board. Both players get three pawns on their row that move normally (barring the double space rule) on their starting row. The goal of the game is to either advance a pawn to the other side of the board, or to prevent the opposing player from moving.

In our project we will be checking to see if white is in a winning state on a given board, and if they are not then we will check how many possible move they are able to make on there turn.



# Propositions

Let x, y be within {1, 2, 3}

w[y][x] is defined as being a white pawn on coordinates (x, y)

b[y][x] is defined as being a black pawn on coordinates (x, y)

o[y][x] is defined as being no pawn on coordinates (x, y)

s[y][x] is defined as a black pawn being stuck on coordinates (x, y)

# Constraints

White pawn can capture by moving to the left:

White pawn can capture by moving to the right:

White pawn can move forward:

White has a winning pawn by moving to the other side of the board:

Black pawn is stuck since his pawn has been blocked:

Black can't move (and has 3 pawns):

Black can't move (and has 2 pawns):

Black can't move (and has 1 pawn):

# Model Exploration

We have looked at winning configurations where a white pawn reaches the other end of the board and whether black can be blocked from making a move on there turn.

We have also looked at the configurations where the number of possible moves is counted.

We have made 3 jape proofs for the project. One for if 1, 2 or 3 black pawns cannot move, then White wins. Another show that a white pawn cannot move if there is not a pawn it can capture and no empty space for it to move to. And the final one Proves if there is no pawn on a square then it is an empty square.

# First-Order Extension

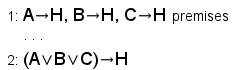
Extending our project into a predicate logic setting would simplify it, our propositions for “Black pawn has no moves” and each “Black can’t move” propositions could be simplified into the following proposition:

Black can’t move:

Using this proposition would completely remove the need for the s proposition.

All other propositions would stay the same however.

# Jape Proofs



* A, B, and C are simplified versions of the positions of the black pawn cannot move.
* H is when white has won the game.
* (This one might be a little to simple)

# 

* P represents a white pawn.
* A represents a black pawn that can be diagonally taken leftwards
* C represents a black pawn that can be diagonally taken rightwards
* B represents an empty space in front of the white pawn
* D represents the ability for the white pawn to move
* The proof would show that a white pawn cannot move if there isn't a pawn it can capture and no empty space for it to move to.



* B represents a black pawn on a given square
* P is a white pawn
* E is an empty square
* This proves that if there is no pawn on a square then it’s an empty square

# Requested Feedback

**Setting up our variables and theory (run.py)-**

We start off our code by creating a board (Line 14-17) then add constraints to the variables b, w, and o (b = Black, w = White, o = empty) based on whether there’s a coloured pawn there or not. We are not sure if this was the right way to setup our theory and do not really want to commit before we get another opinion.

**Counting Number of Moves-**

The part where we are running into the most issues is counting the number of possible moves given a specific board. We are not entirely sure the best way to code it into our project.

**Jape Sequences-**

Coming up with Jape Sequences for this project is seeming to be difficult. Especially because each of our constraints have x and y coordinates where they can be. It is possible to use or , but coming up with the proofs is a difficult feat.