## Project Summary

Sequence is a board game in which players compete to make a sequence of 5 tiles on the board. Each tile corresponds to a card in a standard desk, and occupation is determined by the color of the counter placed on it. Players take turns placing counters using a hand of drawn cards.

The model will correspond to the positions of a player's counters and confirm if a winning scenario is possible (if the player has the cards needed to make a sequence in their hand). If there is no such model, then there is no winning scenario.

Our model will need to consider all the winning and losing scenarios. Will need to check if the player has made a valid move, if there is a winning scenario at any one turn and the type of winning scenario. Our model will use propositional logic to determine the status of the pervious statements.

For the sake of this model, the player being tracked will use blue counters. Propositions

Start position is the top left of the board, the coordinate i increases as you move down the board while j increases as you move to the right.

- $O\_red_{ij}$: True when a tile at position (i, j) has a red counter on it.
- O_blue: True when a tile at position (i, j) has a blue counter on it.
- O_green: True when a tile at position (i, j) has a green counter on it.
- $s\_vertical_{ij}$: True when 5 adjacent vertical tiles have been occupied by the same color, indicating a sequence, (i, j) represents the top
- $s\_horizontal_{ij}$: True when 4 adjacent horizontal tiles have been occupied by the same color, indicating a sequence, (i, j) represents the leftmost tile
- $s\_diagonal\_up_{ij}$: True when 4 tiles have been occupied by the same color, going up from the leftmost position (i, j)
- $s\_diagonal\_down_{ij}$: True when 4 tiles have been occupied by the same color, going down from the leftmost position (i, j)
- $Playable\_red_{ij}$: True if the red player has a card corresponding to the tile, allowing them to place a counter on it
- Playable_blue: True if the blue player has a card corresponding to the tile, allowing them to place a counter on it
- Playable_green: True if the green player has a card corresponding to the tile, allowing them to place a counter on it
- $can\_sequence\_v_{ij}$ /$can\_sequence\_h_{ij}$ /$can\_sequence\_d\_up_{ij}$ /$can\_sequence\_d\_down_{ij}$: Checks if player can make a sequence by seeing if 3 of any 4 adjacent tiles are occupied by blue, and if the remaining tile can be occupied by blue
- $Occupied_{ij}$ : True when a tile at position (i, j) has a counter of any colour on it

## Constraints

- A winning scenario is described as a scenario in which the player can create a sequence
    - $(s\_vertical_{ij} \vee s\_horizontal_{ij} \vee s\_diagonal\_up_{ij} \vee s\_diagonal\_down_{ij})$

- If a tile can be placed at a position, it must not already be occupied, and you must have the appropriate card for the position
    - $(\neg \text{Occupied}_{ij} \wedge \text{can\_occupy}_{ij})$
- Sequences and possible sequences do not overlap with each other
    - $(\neg \text{s\_vertical}_{32} \wedge \text{s\_vertical}_{22})$
- Propositions for sequences only hold when there are blue counters present
    - $(\text{o\_blue}_{ij} \vee \text{o\_red}_{ij} \vee \text{o\_green}_{ij})$
- Player has appropriate card and is can play a token at that position
    - $(\text{playable}_{ij} \wedge \text{playable\_red}_{ij})$
- No token or colour on the specified position
    - $(\neg \text{Occupied}_{ij})$
- Position is already occupied by a red token
    - $(\text{Occupied}_{ij} \wedge \text{o\_red}_{ij})$

## Model Exploration

- Refactoring of Propositions:
  Previously, propositions definitions were complicated, making it hard to work with them. Now, however, we made a common base called "Hashable" that all propositions can use. This makes propositions easier to create and manage, and it helps with set operations and comparisons, ensuring each proposition is unique.
- Updated Card Assignment:
  In previous versions of the code, assigning cards required dealing with various conditions and nested loops. The updated code streamlines this card assignment process by utilizing set operations and distinct pairs, which not only enhances efficiency but also makes it easier to comprehend. This modification decreases the likelihood of duplicate card assignments and enhances the overall reliability of the model.
- During the exploration of our model, we started with an enum to represent players, this ended up being a poor approach so we decided to use separate propositions and constraints. This was much simpler, using the operations o_red, o_green, o_blue, and composing restraints based on how each color would interact with other tiles. This made the model more flexible and comprehensible, overall felicitating greater expansion and stability to the model. Using the Hashable class, and by overriding base methods to compare classes, they became more compatible with each other.
- We previously tried to hard code values, but now make use of loops and random numbers to generate a unique game state with differing tile occupancies and hands of cards, which shows the flexibility of the model.
- Card assignment previously had a convoluted process, which was fixed by introducing unique pairs to minimize card duplicates
- The model is also more precise as it removes redundant parts, focusing on essential points and leaving out unnecessary complexity
- Changes have significantly improved the durability of the model, as it is more flexible and open to expansion. Allowing for a more concise and adaptable representation of the problem.

- Changed the definition of a winning scenario from “a player can make a sequence in the next turn” to “a player has the cards needed to make a sequence”, this makes it possible to explore beyond just a single move, and is more practical since it makes full use of the provided data to represent the bigger picture
- Instead of just evaluating the game state from a single player’s perspective, the model evaluates for blue, red, and green, providing more data about the state of the game without needing any more data than what was given. The model maintains simplicity, but yields much more information

## Jape Proof Ideas

*List the ideas you have to build sequents & proofs that relate to your project.*

- *If a player has a card that can occupy a non-occupied tile, which is adjacent to a line of 4 tiles, which are occupied by the same player. Then can place their own card to make a sequence*
    - $can\_occupy_{ij} \land \neg Occupied_{ij} \land (can\_sequence\_v_{ij} \lor can\_sequence\_h_{ij} \lor can\_sequence\_d\_up_{ij} \lor can\_sequence\_d\_down_{ij}) \rightarrow (s\_horizontal \lor s\_vertical_{ij} \lor s\_diagonal\_up \lor s\_diagonal\_down_{ij})$
- *If a player has a card that can occupy a non-occupied tile, and that tile is not adjacent to a line of 4 tiles which are occupied by the same player, then they cannot make a sequence by placing the tile.*
    - $can\_occupy_{ij} \land \neg Occupied_{ij} \land (c \neg an\_sequence\_v_{ij} \lor \neg can\_sequence\_h_{ij} \lor \neg can\_sequence\_d\_up_{ij} \lor \neg can\_sequence\_d\_down_{ij}) \rightarrow (\neg s\_horizontal \lor \neg s\_vertical_{ij} \lor \neg s\_diagonal\_up \lor \neg s\_diagonal\_down_{ij})$
- If a player cannot occupy a tile that is adjacent to line of 4 tiles that are occupied by the same player, then they cannot make a sequence.
    - $\neg can\_occupy_{ij} \land \neg Occupied_{ij} \rightarrow (\neg s\_horizontal \lor \neg s\_vertical_{ij} \lor \neg s\_diagonal\_up \lor \neg s\_diagonal\_down_{ij})$

## Requested Feedback

*Provide 2-3 questions you’d like the TA’s and other students to comment on.*

1. Any ideas on how to implement multiple encoders into our code, with the goal of having a separate encoder for each possible win-case?
2. If two sequences were created at the same time, for example a vertical sequence and horizontal sequence, are there any edge cases in the logic that we missed?
3. Were all constraints and propositions considered? Were there any edge cases that were missed? Are any of them repetitive or unnecessary?

## 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!***

For all 4 adjacent occupied tiles of the same color in a line(horizontal, vertical, diagonal) and one of the ends is unoccupied, there exists a sequence.

$\forall$ (can_sequence_$v_{ij}$ $\vee$ can_sequence_$h_{ij}$ $\vee$ can_sequence_d_up$_{ij}$ $\vee$ can_sequence_d_down$_{ij}$) $\rightarrow$
$\exists$ (s_horizontal: s_vertical$_{ij}$: s_diagonal_up: s_ diagonal_down$_{ij}$ )

Using the existential quantifiers, and defining the sequences as functions with 2 variables (i and j), we have a more robust and simple way of expressing the existence of a sequence:

$\exists$i. $\exists$j. (s_horizontal(i, j) $\vee$ s_vertical(i, j) $\vee$ s_diagonal_up(i, j) $\vee$ s_diagonal_down(i, j)