# Project Summary

*Short summary of the project setting.*

In our project, we aim to model the two-player strategy board game, Battleship, and solve a way to cheat, if we were allowed to move some of the ships. A fleet of ships is placed onto a grid. To steer our project in a unique direction, we will explore how players can shift their ships in response to incoming shots to avoid being hit. It is considered a miss if the opponent incorrectly targets a position without a ship, or if the player successfully shifts their ship out of harm’s way. A shot is a hit if the opponent correctly identifies the location of a ship, and the player has no option to move their ship to avoid the hit. Our model will use logical propositions and constraints to represent the game board, ship placements and shifts, and shots fired systematically.

# Propositions

*List of the propositions used in the model, and their (English) interpretation.*

*Note: Y can be either 1 or 2, representing either player 1’s board state/ships or player 2’s.*

1. **Ship Propositions:**

* **shipY\_xij(t):** True if position (i, j) on the grid is occupied by ship x at time t.
* **bumpedY\_xij(t):** True if ship x has been moved to an adjacent position (bumped) from its original position at time t.
* **shipY\_movable\_x(t):** True if ship xcan be moved to a new position without violating game constraints at time t.

1. **Shot Propositions:**

* **shotY\_ij(t):** True if position (i, j) has been shot (fired at) by a player at time t.
* **hitY\_ij(t):** True if position (i, j) was a hit at time t, meaning that a ship occupies the position and cannot be moved to avoid the shot.
* **missY\_ij(t):** True if position (i, j) was a miss at time t, meaning that the shot was fired at an empty position or the ship successfully shifted away.

1. **Sunk Ship Proposition:**

* **sunkY\_x(t):** True if ship xhas been sunk at time t.
* **game\_over(t):** True if all ships of a player have sunk.

1. **Turn Indicators:**

* **turn1(t):** True if it is player 1's turn at time t.
* **turn2(t):** True if it is player 2's turn at time t.

# Constraints

*List of constraint types used in the model and their (English) interpretation. You only need to provide one example for each constraint type: e.g., if you have constraints saying “cars have one colour assigned” in a car configuration setting, then you only need to show the constraints for a single car. Essentially, we want to see the pattern for all of the types of constraints, and not every constraint enumerated.*

1. Ship Placement Constraints:
   * No two ships of the same player can occupy the same cell:
     + ¬(ship1\_xij(t) ∧ ship1\_yij(t))
   * A ship can occupy multiple cells: e.g. If a carrier occupies 3 cells:
     + ship2\_carrierA1(t) ∧ ship2\_carrierA2(t) ∧ ship2\_carrierA3(t)
2. Shot Constraints:
   * A shot is either a hit or miss:
     + shot1\_A1(t) → (hit1\_A1(t) ∨ miss1\_A1(t))
   * A shot is a hit if there is a ship there:
     + shot1\_B5(t) ∧ carrier1\_B5(t) → hit1\_B5(t)
   * A shot is a miss if there is no ship there or if the ship has been successfully shifted away:
     + shot1\_C6(t) → (¬ship1\_C6(t) ∨ bumped\_xC6(t))
3. Bumping Mechanism:
   * Allow ships to be bumped to adjacent positions after each opponent shot (or stay):
     + shot1\_kl(t) → (ship2\_xij(t) ∨ (¬ship2\_y(k+1)(l)(t) ∧ ¬ship2\_y(k-1)(l)(t) ∧ ¬ship2\_y(k)(l+1)(t) ∧ ¬ship2\_y(k)(l-1)(t)))
   * Only one ship bump per turn is allowed (bump one ship based on the opponent’s shot)
     + turn1(t) → (¬turn2(t) ∧ bumped\_xij(t))
   * Ensure that multiple ships cannot occupy the same cell after a bump (a bumped ship does not overlap an already placed ship)
     + bumped1\_xij(t) → ¬(ship1\_xij(t))
4. Turn-Based Constraints
   * Only the player whose turn it is can make a move (shoot or move)
     + turn1(t) → (shot1\_ij(t) ∨ bumped1\_xij(t))
     + turn2(t) → (shot2\_ij(t) ∨ bumped2\_yij(t))
   * After one player takes their turn, it should be the other player’s turn
     + (turn1(t) → turn2(t+1)) ∧ (turn2(t) → turn1(t+1))
   * If a player has no ships remaining (game is over), neither player can take a turn
     + (game\_over(t) → ¬turn1(t) ∧ ¬turn2(t)
5. Game End Condition
   * The game ends when all ships of one player are sunk.
     + (sunk1\_carrier(t) ∧ sunk1\_battleship(t) ∧ sunk1\_destroyer(t) ∧ sunk1\_submarine(t) ∧ sunk1\_patrolboat(t)) → game\_over(t)

# Model Exploration

*List all the ways that you have explored your model – not only the final version, but intermediate  
versions as well. See (C3) in the project description for ideas.*

We started by modelling a basic Battleship setup using logical propositions and constraints to represent the different parts of the game. This includes setting up a grid as the game setting and using logic to track ship placements, hit-and-miss outcomes, and turn-taking. In this setup, each player places their ships onto the grid and takes turns firing shots, to sink all the other player's ships. Each shot targets a specific cell (coordinate), and the outcome depends on whether that cell is occupied by a ship or not.

As we progressed, we decided to pivot to a new concept: the "Battleship cheats" model. This model would allow players to adjust, or "nudge", their ship positions in response to opponent shots - adding a sort of in-game "cheat". To add strategic depth to this mechanic, we explored two main approaches. We considered allowing the opponent to fire multiple shots per turn while the defending player would reposition their ships using the model to minimize the amount of damage taken. However, to keep it from being too complex, we settled on a one-shot-per-turn format, where the defending player can nudge a ship after each shot to avoid a hit.

To prevent the solution from always being "move the ship out of the shot's path," we decided to add several rules and constraints to add more strategic depth to the cheat mechanic.

# Jape Proof Ideas

1. This proposition states that if Player A is taking their turn (TurnATurnA) and has made a shot at position A1 (ShotA1ShotA1), it is equivalent to asserting that Player B cannot take their turn or has not shot at position B1 (¬TurnB∧¬ShotB1¬TurnB∧¬ShotB1). This captures the fundamental rule of turn-based gameplay in Battleship, where only one player can act during their turn, ensuring a structured sequence of actions.
2. This proposition asserts that if Player A has a ship located at position A1 (ShipXShipX) and then takes a shot at A1 (ShotA1ShotA1), the outcome is a confirmed hit at that position (HitA1HitA1). This establishes the basic interaction in the game between a player's shot and the location of their ships. It captures the essence of Battleship, where a successful shot results in a hit if a ship occupies the targeted coordinate.
3. This sequent states that if Player A takes a shot at position B3 during their turn (TurnATurnA), and there is a movable ship XX that is not currently at B3 (¬ShipAtB3¬ShipAtB3), then the result is a miss at B3 (missB3missB3). This captures the dynamic aspect of the game, where ships can move to avoid being hit. It illustrates the strategic element of movement in Battleship, indicating that players can evadeshots by relocating their ships to unoccupied adjacent spaces.

*(In the jape file, ignore the second sequent that just says “ship”, it is formatted incorrectly)*

# Requested Feedback

1. Are the proposed propositions and constraints clear and effective in describing the important game? Specifically, regarding the bumping mechanism, are there any additional constraints that should be included to enhance clarity?
2. Do you think the cheating aspect is well-integrated into the overall game mechanics? Should it be changed so it is not limited to turn by turn (move one ship based on the opponent’s move) and instead be able to cheat by moving multiple ships based on multiple opponent guesses?
3. What are your thoughts on the JAPE proofs provided for modeling the Battleship game mechanics? Specifically, do you find the equivalences clearly defined, and are the explanations effective in conveying the intended relationships between turns, shots, and outcomes? Additionally, are there any areas where you think the proofs could be strengthened or clarified further?

# First-Order Extension

We can extend our model to a predicate logic setting by including predicates that replace the propositions and constraints we may use either in the current framing of our model or in an extension.

**Predicates:**

1. **Ship Propositions:**

* **ShipY(x, i, j, t):** Represents that ship x occupies position (i, j) at time t.
* **BumpedY(x, i, j, t):** Represents that ship x has been bumped to position (i, j) at time t.
* **MovableY(x, y):** Represents that ship x can be moved without violating constraints at time t.

1. **Shot Propositions:**

* **ShotY(i, j, t):** Represents that a shot was fired at position (i, j) at time t.
* **HitY(i, j, t):** True if position (i, j) was a hit at time t.
* **MissY(i, j, t):** True if position (i, j) was a miss at time t.

1. **Sunk Ship Proposition:**

* SunkY(x, t): Represents that ship x has been sunk at time t.
* GameOver(t): True if all ships of a player have sunk at time t.

1. **Turn Indicators:**

* Turn1(t): True if it is player 1’s turn at time t.
* Turn2(t): True if it is player 2’s turn at time t.

**Constraints:**

1. Ship Placement Constraints:
   * No two ships of the same player can occupy the same cell:
     + x1,x2(x1 x2 ShipY(x1, i, j, t) ShipY(x2, i, j, t) )
       - A contradiction occurs when two distinct ships occupy the same position at time t.
2. Shot Constraints:
   * A shot is a hit if there is a ship there:
     + i,j(ShotY(i, j, t) ShipY(x, i, j, t) HitY(i, j, t))
       - The outcome is a hit if a shot is fired at a position that occupies a ship x at time t.
3. Bumping Mechanism:
   * The game ends when all the ships of one player are sunk:
     + x(SunkY(x, t) GameOver(t))
       - When all ships x have been sunk the game is over.