# Project Summary

*The project develops a formal logic model of the card game War, focusing on employing propositional logic to systematically represent and analyze the dynamics of card distribution, gameplay decisions, and winning strategies. By abstracting the game into logical propositions and constraints, the model aims to simulate various game scenarios and to identify key strategic factors that influence the outcome of games, thereby offering insights into optimal gameplay tactics.*

# Propositions

* **Card(rank, suit)**: Represents a card in the deck with a specific rank and suit.
* **Owns(player, card)**: Indicates that a particular player owns a specific card.
* **Plays(player, card, round\_number)**: Represents a player playing a specific card in a given round.
* **Wins(player, round\_number)**: Indicates that a specified player wins a particular round.
* **Tie(round\_number)**: Represents that a tie occurs in a specified round.
* **FinalTie(round\_number)**: Indicates that no winner is found even after 3 tie-breaking rounds, resulting in a final unresolved tie.
* **HigherRank(card1, card2)**: Denotes that card2 has a higher rank than card2.
* **SameRank(card1, card2)**: Denotes that two cards have the same rank.
* **OverallWinner(player)**: Indicates that a specified player wins more rounds overall than the other player.

# Constraints

* **Owns("Player A", c)** *→* **¬Owns("Player B", c)**: A card can only be owned by one player. If Player A owns card c, then Player B doesn’t and cannot own the same card.
* **Plays("Player A", c, r) ∨ Plays("Player B", c, r)**: In each round r, both players must play exactly one card, ensuring that every player makes a move.
* **Plays("Player A", c, r) → Owns("Player A", c)**: A player can only play a card they own. For example, if Player A plays card ccc in round r, they must own that card.
* **Plays("Player A", c, r) ∧ Plays("Player B", c₂, r) ∧ HigherRank(c₁, c₂) → Wins("Player A", r)**: Player A wins round r if they play a card c₁ that has a higher rank than the card c₂ played by Player B.
* **Plays("Player A", c₁, r) ∧ Plays("Player B", c₂, r) ∧ SameRank(c₁, c₂) → Tie(r)**: A round r ends in a tie if both players play cards of the same rank.
* **Wins("Player A", r) ∨ Wins("Player B", r) ∨ Tie(r));** **¬(Wins("Player A", r) ∧ Wins("Player B", r))**: Only one outcome can occur in each round: either Player A wins, Player B wins, or there is a tie. It is impossible for both players to win the same round.
* **Tie(r) → (Plays("Player A", c₁, r+1) ∧ Plays("Player B", c₂, r+1))**: If round r ends in a tie, a tie-breaker round occurs. Both players must play one card in each tie-breaker round, with up to three additional rounds allowed.
* **¬Wins("Player A", r) ∧ ¬Wins("Player B", r) → FinalTie(r)**: If neither player wins after the allowed tie-breakers, the round is declared a final tie.

# Model Exploration

We started by establishing the fundamental rules of War, beginning with card ownership. We defined propositions like Owns("Player A", x) and Owns("Player B", x), ensuring that each card was exclusively owned by one player. This allowed us to simulate the initial card distribution and gave us a starting point for modeling each player's moves. From there, we moved on to gameplay actions. We introduced propositions such as Plays("Player A", x, r) and Plays("Player B", x, r), representing when each player plays a card in a given round, tracking who plays what card and when. Once the core gameplay was represented, we focused on winning conditions based on card ranks. We added propositions like HigherRank(x, y) to identify when one card outranked another, determining the round winner. Additionally, we incorporated SameRank(x, y) to handle ties, which meant further rounds would be needed to break them.

We had deeper exploration for the model by exploring stacked deck scenarios where a player was guaranteed to win all rounds. We set up a deck where Player A had the highest-ranked cards for each round, ensuring deterministic outcomes. This allowed us to test whether the model could reliably simulate a scenario where Player A won every single round, reflecting the extreme impact of strategic deck stacking. The stacked deck simulation revealed that the model could accurately represent a scenario where one player consistently won, reinforcing our understanding of how War’s gameplay can be influenced by card distribution and rank advantages. This helped us see the potential of the model to simulate not just standard gameplay but also highly strategic or rigged scenarios.

Another critical part of our exploration was ensuring the model could handle tie scenarios effectively. We implemented a system for up to three tie-breaker rounds, which was necessary to reflect the real complexity of War’s tie-breaking rules. In these situations, each player drew additional cards, and the model needed to check if one of the players had a higher-ranked card to resolve the tie. We tested this by setting up situations where ties were forced in specific rounds, allowing us to see whether the model correctly triggered tiebreakers and managed persistent ties. This exploration confirmed that the model could handle both simple ties and more complex multi-round tie-breaking situations.

Were still working on exploring partial assignments where Player A was set to win at least 70% of the rounds. By adding constraints that forced Player A to win a minimum number of rounds, we observed how the model adjusted to meet this condition. This will help us understand the model’s flexibility and its ability to handle specific win targets. The model successfully simulates realistic gameplay, tie-breaking processes, and deterministic outcomes through stacked decks. At this stage, we’re pleased with the model’s ability to simulate basic gameplay dynamics with a strong foundation for exploring more complex strategies in future iterations.

# Jape Proof Ideas

1. *If for every card x that Player A owns, Player B cannot own the same card, then it logically follows that there is no card x that both Player A and Player B own simultaneously.*

A screenshot of a computer

Description automatically generated

*Premise: (OwnsA(x) → ¬OwnsB(x))*

*Conclusion: ¬ (OwnsA(x) ∧ OwnsB(x))*

1. Premise:

**Premise 1**: PlaysA(c,r) → OwnsA(c)– If Player A plays card c in round r, then Player A owns that card.

**Premise 2**: WinsA(r) → PlaysA(c1,r) ∧ PlaysB(c2,r) ∧ HigherRank(c1, c2 – Player A wins round r if Player A plays c1, Player B plays c2, and c1​ is higher-ranked than c2.

**Premise 3:**  HigherRank(c,c′)→ ¬ SameRank(c,c′) – If a card c has a higher rank than another card c′, then they cannot have the same rank.

A screenshot of a computer

Description automatically generated

Conclusion: If Player A wins round r, then Player A owns a card that is not of the same rank as Player B’s card

1. Premises:

**Premise 1**: PlaysA(c,r)→OwnsA(c): If Player A plays card c in round r, then Player A must own that card.

**Premise 2**: OwnsA(c)→¬OwnsB(c): If Player A owns card c, then Player B cannot own that card.

A screenshot of a computer

Description automatically generated

Conclusion: if Player A plays card c in round r, then Player B does not own that card.

# Requested Feedback

1. How complex do we need our proofs?
2. how we're supposed to show our implementation of advanced scenarios.
3. Predicate notation: Is it clear how to transition from propositional constraints to quantified expressions?
4. Do we have a good representation of a tie scenerio?

# First-Order Extension

To extend our War game model to a predicate logic setting, we would generalize our propositions and constraints to include parameters and quantifiers, enabling a more flexible and scalable representation of the game’s rules and conditions. Instead of specific propositions, we would use parameterized predicates, allowing us to define relationships and properties across all rounds, cards, and players. This extension would use quantifiers (∀ for "for all" and ∃ for "there exists") to express game rules more generally and avoid the need for enumerating specific cases.

For example, our proposition Card(rank, suit) could remain mostly the same, but it would serve as a parameterized entity in conjunction with other predicates rather than a standalone proposition. In predicate logic, Owns(player, card)would allow us to express ownership relationships with quantifiers. We could represent exclusive ownership by defining a constraint like ∀c (Owns("Player A", c) → ¬Owns("Player B", c)), which asserts that if Player A owns a card, Player B cannot own the same card. This constraint would ensure mutual exclusivity in ownership for all cards without needing to specify each one individually.

The Plays(player, card, round\_number) predicate would become more expressive in predicate logic, allowing us to ensure that each player plays exactly one card per round across all rounds. We could set constraints such as ∀r ∃!c (Plays("Player A", c, r)) ∧ ∃!c (Plays("Player B", c, r)), ensuring that both players play exactly one card in each round. Additionally, we would define constraints like ∀p, c, r (Plays(p, c, r) → Owns(p, c)) to ensure that a player can only play cards they own.

The winning conditions constraint such as ∀p, r (Wins(p, r) ↔ (∃c1, c2 (Plays(p, c1, r) ∧ Plays(other\_player(p), c2, r) ∧ HigherRank(c1, c2)))) could represent that a player wins a round if they play a higher-ranked card than their opponent’s card. Similarly, the Tie(round\_number) predicate could be defined as ∀r (Tie(r) ↔ (∃c1, c2 (Plays("Player A", c1, r) ∧ Plays("Player B", c2, r) ∧ SameRank(c1, c2)))), which would ensure that a tie occurs if both players play cards of the same rank in any round.

# Useful Notation

*Feel free to copy/paste the symbols here and remove this section before submitting.*