**Three Cards and a River Poker**

Group 23: Luke Bailey, Connor Pineault, Cameron Jenkins

**Section 1: Project Summary**

Our project strives to model a simplified version of poker. This version includes 3 cards and 2 shared table cards (the river). The model determines the rank of each player's hand. Then, the model uses those ranks to determine which hand wins the round. The model also provides suggestions on if the player should play the hand or fold according to their hand.

Rules:

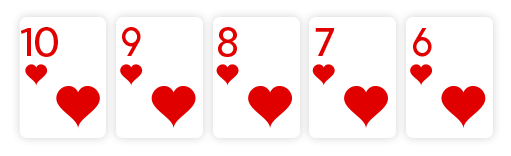
The game consists of two players: a user and the dealer (CPU). Also, the game is played with a standard 52-card deck. The objective of the game is for the player to beat the dealer's hand by having a better hand. These are the steps of the gameplay:

1. Three cards are dealt to the player and three cards are dealt to the dealer. The cards are dealt face down on the table.
2. Next, two cards are dealt face up on the table. These two cards are shared between the user and dealer. You can think of them as being “added '' to each person's hand such that both players have a hand of five cards.
3. After the cards have been dealt, the player is able to look at the three cards dealt to them.
4. The player uses the three cards in their hand and the two table cards to determine whether they want to play or fold the hand. If the player decides to play the hand, the dealer must also play the round. Both hands are revealed and the winner is determined based on who has the better hand. Conversely, if the player decides not to play the hand, the dealer automatically wins the round
5. The player communicates their decision to play or fold the hand
6. The winner is determined (Note: In our version, if both players have hands of the same ranking and the same high card in their hand, then the dealer wins)

As previously mentioned, the winner of the round is the player with the best hand. The order of the hand rankings in our game is:

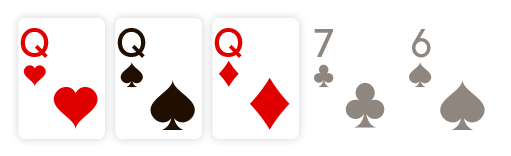
1. Straight flush: five cards of the same suit and of sequential rank

Ex.



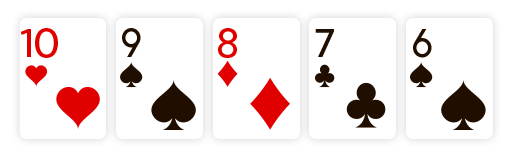
1. Three of a kind: three cards of the same number

Ex.



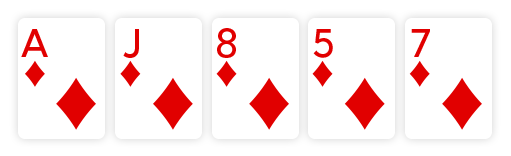
1. Straight: five cards of sequential rank

Ex.



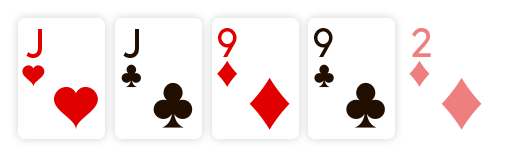
1. Flush: five cards of the same suit

Ex.



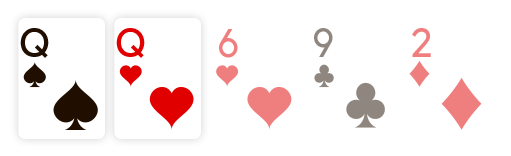
1. Two-Pair: 2 sets of Pairs

Ex.



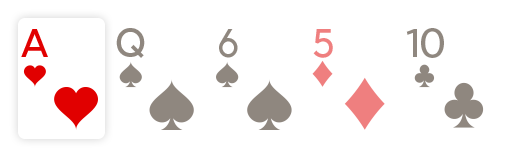
1. Pair: two cards of the same number

Ex.



1. High card: no other ranking is satisfied, so the hand is evaluated by its highest number.

Ex.



**Note: The high card comes from the player's 3-card hand and not the shared table cards!**

**Note**: These rankings may differ from other versions of poker but are the ones we adapted for our game. Also, the previous 7 images were retrieved from https://www.888poker.ca/how-to-play-poker/hands/.

We will be modeling this game by using logic to evaluate what hand rank a given hand evaluates to. Also, we will be using logic to compare two hands and determine which is better. We also used logic to create multiple models that recommend whether a player should play or fold a given hand using the three cards they were dealt and the two shared table cards. We reduced the scope of the rankings by removing ranks that are often used for five-card poker such as four of a kind, royal flush, and full house. These were removed because we originally were planning on modeling three card poker but later adjusted our game to three cards and a river. We did this because the shared cards increased the complexity of our models but we did not have enough time left to add every new rank. Furthermore, there are no reductions in the scope of the deck and we are utilizing a standard 52-card deck.

Finally, our game consists of two modes: gameplay and model accuracy. The first mode is the gameplay which allows the user to play as many rounds as they want. The program outputs the users three cards and the shared table cards. Also, the program evaluates the user's hand and outputs its rank and a play/fold recommendation. Then, the program requests input from the user on whether they want to play or fold each round. Next, the program reveals the dealers cards and announces which player won the round. Additionally, the program outputs the user’s play or fold decision accuracy as well as the accuracy of our simplest model at the end of each round. Furthermore, at the end of any round the user can exit the gameplay mode and enter the model accuracy mode. In this mode, the user can test out the accuracy of different models we developed that decide whether to play or fold each hand (See model exploration for more details on this mode).

**Section 2: Propositions**

Before setting up the propositions we must model the cards and the hand:

Consider that a players hand consist of five total cards (3 individual + 2 shared):

Each card has two ideas that must be modeled: suit and number

**Note:** We order cards from lowest to highest in hand

Let i be the card. Users hand is made of five cards: (card0, card1, card2, card4,

card5)

Cardi will have a suit and a number

Consider that there are two players:

Let j be the player number. There are two players: (H0 and H1)

If not specified, assume that 0 is the user and 1 is the dealer

We model the suit as:

C is true if the card is a club

D is true if the card is a diamond

S is true if the card is a spade

H is true if the card is a heart

Ex. C0 is true if first card in hand is clubs

We model the number as:

2 is true if the card is a 2

3 is true if the card is a 3

4 is true if the card is a 4

5 is true if the card is a 5

6 is true if the cards is a 6

7 is true if the card is a 7

8 is true if the card is a 8

9 is true if the card is a 9

10 is true if the card is a 10

11 is true if the card is a jack

12 is true if the card is a queen

13 is true if the card is a king

14 is true if the card is an ace

Ex. 81 is true if second card in hand is an 8

We also require propositions for the rank associated with the hand:

-SF is true if the hand is a straight flush

-T is true if the hand is a three of a kind

-S is true if the hand is a straight

-F is true if the hand has a flush

-TP is true if hand has a two pair

-P is true if the hand has a pair

-H is true if the hand is a high card

Ex. If a hand has a straight flush, then SF will be true

High card comparator:

We require a high card comparator proposition to keep track of which hand has the better high card:

Let HCC0 be true if the high card of the first player/user's hand is higher than the high

card of the second player/dealers hand

Win:

We also require a proposition to determine the winner

W0 is true if the first player (user) wins the round

Hand recommendations:

We require a recommendation proposition to capture whether a user should play or fold a hand

RP is true if the model recommends that the user should play

~RP is true if the model recommends that the user should fold

**Section 3: Constraints**

We use constraints to model three different aspects of the game: hand ranking constraints, winning constraints, and recommendation constraints

**First, we have the following constraints that determine the hand ranking.:**

**Note:** We order cards from lowest to highest in hand

Let i be the card. Users hand is made of five cards: (card0, card1, card2, card4, card5)

Straight Flush:

Ex. If all five cards are clubs and the five cards are 2, 3, 4, 5, 6, then the hand is a straight flush

((C0 ∧ C1 ∧ C2 ∧ C3 ∧ C4) ∧ (20 ∧ 31 ∧ 42  ∧ 53 ∧ 64)) → SF

Note: Evaluate for all suits and for all numbers (2-13) of those suits

Three of a Kind:

Ex. If three cards are two’s, then the hand is a three of a kind:

((20 ∧ 21 ∧ 22) ∨ (21 ∧ 22 ∧ 23) ∨ (22 ∧ 23 ∧ 24)) → TK

Note: Evaluate for all numbers 2-13

Straight:

Ex. If the five cards are 2, 3, 4, 5, 6, then the hand is a straight:

(20 ∧ 31 ∧ 42  ∧ 53 ∧ 64) → S

Note: Evaluate for all numbers 2-13

Flush:

Ex. If all five cards are clubs, then the hand is a flush:

(C0 ∧ C1 ∧ C2 ∧ C3 ∧ C4) → F

Note: Evaluate for all suits (C, D, H, S)

Two Pair:

Ex. If two cards are two’s and two cards are five’s, then the hand is a pair:

(((20 ∧ 21) ∧ (52 ∧ 53)) ∨ ((20 ∧ 21) ∧ (53 ∧ 54)) ∨ ((21 ∧ 22) ∧ (53 ∧ 54))) → TP

Note: Evaluate for all numbers 2-13

Pair:

Ex. If two cards are two’s, then the hand is a pair:

((20 ∧ 21) ∨ (21 ∧ 22) ∨ (22 ∧ 23) ∨ (23 ∧ 24)) → P

Note: Evaluate for all numbers 2-13

High Card:

If the hand is none of the higher ranks, then the hand is a high card:

(~SF ∧ ~TK ∧ ~S ∧ ~F ∧ ~TP ∧ ~P) → HC

**Second, we have the following constraints to determine the winning hand:**

Straight Flush

User has a straight flush, the dealer does not. User wins:

(SF0 ∧ ~SF1) → W0

Both players have a straight flush, and the user has a better high card. User wins:

(SF0 ∧ SF1 ∧ HCC0 ) → W0

Both players have a straight flush, and the user has a better high card. User loses:

(SF0 ∧ SF1 ∧ ~HCC0) → ~W0

Three of a kind

User has three of a kind, the dealer has worse than three of a kind. User wins:

(T0 ∧ ~T1 ∧ ~SF0) → W0

Both players have a three of a kind, and the user has a better high card. User wins:

(T0 ∧ T1 ∧ HCC0) → W0

Both players have a three of a kind, and the user has a better high card. User loses:

(T0 ∧ T1 ∧ ~HCC0) → ~W0

Straight

User has a straight, the dealer has worse than a straight. User wins:

(S0 ∧ ~S1 ∧ ~SF1 ∧ ~T1) → W0

Both players have a straight, the user has a better high card. User wins:

(S0 ∧ S1 ∧ HCC0) → W0

Both players have a straight, the user has a better high card. User loses:

(S0 ∧ S1 ∧ HCC0) → ~W0

Flush

User has a flush, the dealer has worse than a flush. User wins:

(F0 ∧ ~F1 ∧ ~S1 ∧ ~SF1 ∧ ~T1) → W0

Both players have a flush, the user has a better high card. User wins:

(F0 ∧ F1 ∧ HCC0) → W0

Both players have a flush, the user has a better high card. User loses:

(F0 ∧ F1 ∧ ~HCC0) → ~W0

Two Pair

User has two pairs, the dealer has worse than two pairs. User wins:

(TP0 ∧ ~TP1 ∧ ~S1 ∧ ~SF1 ∧ ~T1 ∧ ~F1) → W0

Both players have two pairs, and the user has a better high card. User wins:

(TP0 ∧ TP1 ∧ HCC0) → W0

Both players have two pairs, and the user has a better high card. User loses:

(TP0 ∧ TP1 ∧ ~HCC0) → ~W0

Pair

User has a pair, the dealer has worse than a pair. User wins:

(P0 ∧ ~P1 ∧ ~F1 ∧ ~S1 ∧ ~SF1 ∧ ~T1) → W0

Both players have a pair, and the user has a better high card. User wins:

(P0 ∧ P1 ∧ HCC0) → W0

Both players have a pair, and the user has a better high card. User loses:

(P0 ∧ P1 ∧ HCC0)) → ~W0

High card

Both hands are ranked high card, the user has a better high card. User wins:

(H0 ∧ H1 ∧ HCC0) → W0

Both hands are ranked high cards, and the user has a worse high card. User wins:

(H0 ∧ H1 ∧ ~HCC0) → ~W0

Dealer Win

If the first user does not win by any of the previous cases, then the dealer wins:

(~W0 → W1)

**Third, we have constraints for the play or fold recommendation:**

If the user has a hand that is not a high card, then recommend play:

~HC0 → RP0

If the user has a hand that is a high card, then recommend fold:

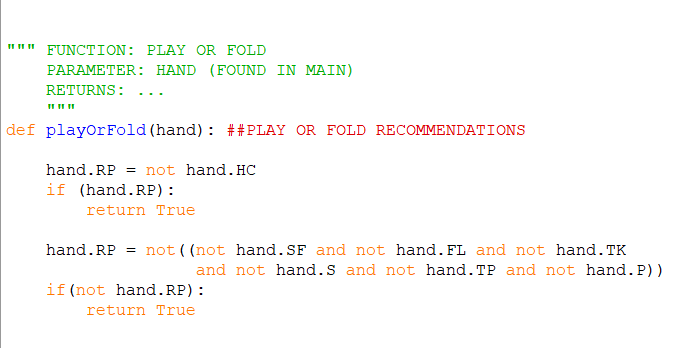
(~**(SF0** ∨ TK0 ∨ S0 ∨ FL0 ∨ TP0 ∨ **P0**)) → ~RP0

**Note**: This is our simplest constraint for the models play or fold recommendation. For more information on play or fold recommendations go to the “Recommendation Model Accuracy” found in the “Model Exploration” section of this report.

**Section 4: Model Exploration**

1. **Recommendation Model Accuracy: Pair Plus Model**

We decided to implement code that kept track of the accuracy of our recommendations. We started with the simple constraints that we would recommend the user plays when they have a pair or better. Conversely, we recommended the user folds when they do not have a pair or better.



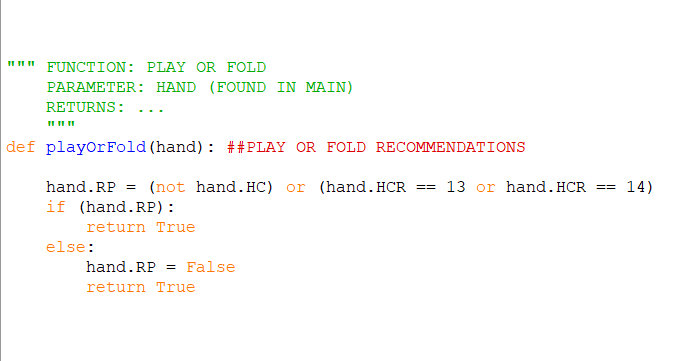
We calculated its accuracy for 100,000 rounds and the model was correct 73.84% of the time. We felt this accuracy was fairly good. However, we wanted to adjust these constraints and try to find a model with better accuracy. There are two more examples of models we evaluated in later sections of this report.

We implemented a function into our main code that allows for the testing of the model accuracy. You can get to this mode by pressing any key at the end of the round to exit the standard gameplay. This function is passed the model of your choosing and outputs the model’s accuracy at making correct fold/play decisions. If the model determines the user should play and the user wins if they would have played this is counted as a correct decision. If the model determines the user should fold and the user would have lost if they played then this is a correct decision. If the model determines that the user should play and the user would have lost if they played then this is not counted as a correct decision. Finally, if the model determines that the user should fold and the user would have won if they played then this is a correct decision. We determined that the model can not be 100% accurate because there are many unknown factors but we wanted to develop a model that made the best decisions possible. For example, if the user has a flush, then the model will recommend that the user plays. However, if the dealer had a straight flush then the user will lose. This would be counted as an incorrect decision but it is still the best decision with the information the model was given. Therefore, our goal was to develop the model that makes the correct decision at the highest rate knowing it would never be 100%.

1. **Recommendation Model Accuracy: Pair Plus and High Card Model**

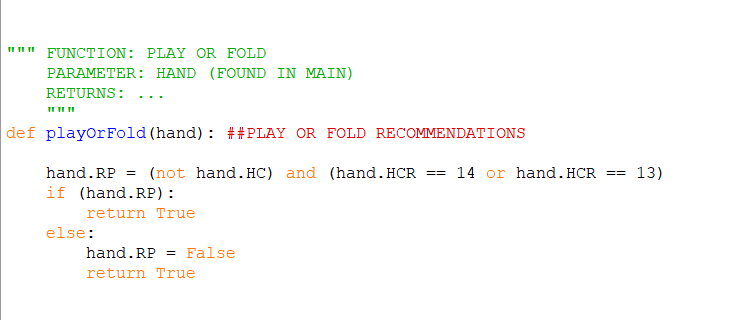
In the next model we developed, we wanted to consider how the high card that is with the pair affected the probability of winning. We felt this was an interesting aspect to test because ties are settled by who has the better high card.

To start, we tested how the model performed if the model recommended playing when the user had a hand that was not a high card or if the high card was 13 (king) or 14(ace).



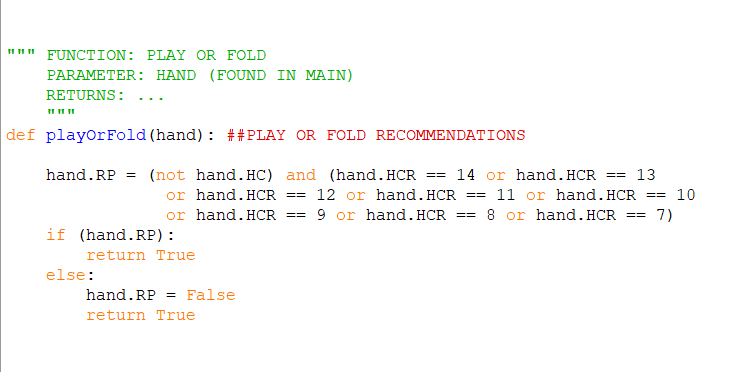
We calculated its accuracy for 100,000 rounds and the model was correct 67.18% of the time. Therefore, we decided to add some further constraints to help improve our model’s recommendation accuracy.

The next constraint we tested was if the user had a hand that was rank pair or higher, and the hand’s high card was an ace or king.



We calculated its accuracy for 100,000 rounds and the model was correct 65.44% of the time. Therefore, we decided to add some further constraints to help improve our model’s recommendation accuracy. We realized this model was likely not accurate because it was recommending to fold on hands with at least a pair because the high card outside the pair was not very high. For example, if your hand was a 5, 5, 5, 4, 3 then it would tell you to fold because the hand did not have a good high card even though it was a three of a kind which has a high probability of winning against the dealer’s hand.

Therefore, we decided to continue investigating the pair plus high card model to determine whether it was even a good idea to involve the high card. We decided to next constraint we tested was if the user had a hand that was rank pair or higher, and the hand’s high card was a 7 or better (ace, king, queen, jack, 10, 9, 8, or 7)



**Note:** This function is named “playOrFold2” in the code

We calculated its accuracy for 100,000 rounds and the model was correct 73.78% of the time.

Finally, we decided to investigate further if the high card was a good factor to look at by examining models with pair plus and ace, pair plus and king plus, etc. The results are displayed in the following table:

| **Pair and high card…** | **Model Accuracy** |
| --- | --- |
| Ace | 60.59% |
| King or higher | 65.67% |
| Queen or higher | 68.89% |
| Jack or higher | 70.92% |
| 10 or higher | 72.02% |
| 9 or higher | 72.71% |
| 8 or higher | 73.53% |
| 7 or higher | 73.53% |
| 6 or higher | 73.86% |
| 5 or higher | 74.12% |
| 4 or higher | 73.91% |
| 3 or higher | 74.03% |

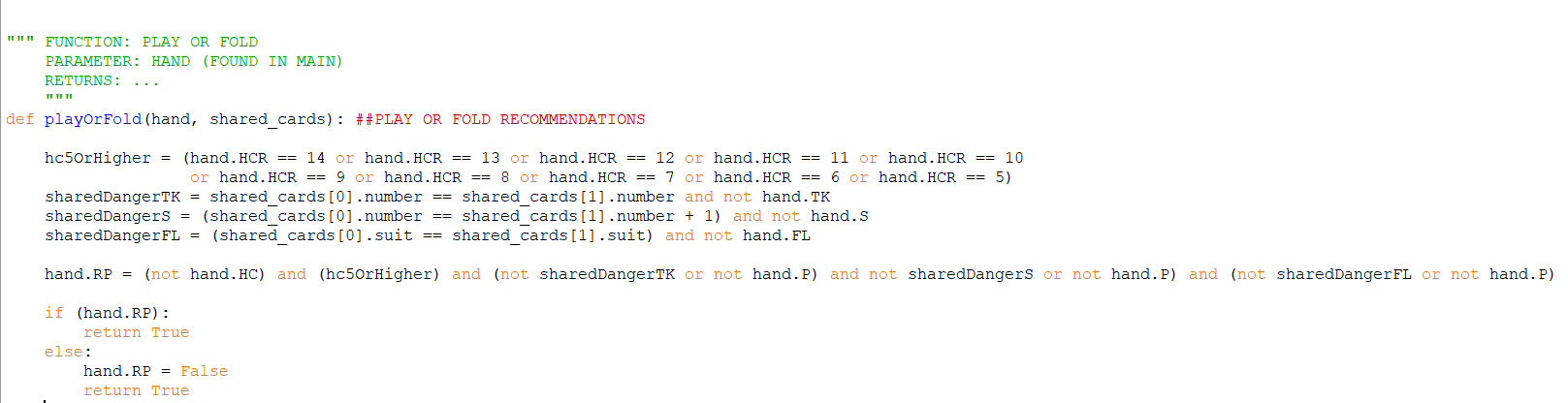
Therefore, we decided to select the constraint of 5 or higher for the pair plus high card model because it had the best accuracy. This is likely because there are less solutions that lose because you folded due to a low high card even though you would have won (ex. 2, 2, 3, 3, 4), then solutions that lose because both players have a hand of the same rank (likely pair) but the other player's high card is better.

1. **Recommendation Model Accuracy: Pair Plus, 5 High, and Shared Danger Model**

Lastly, we decided to consider the shared cards to see if that would improve the models accuracy. We used the previous model of pair and a high card of 5 or higher as a starting point because it had the highest accuracy from the table above. Then, we considered when the user's hand may be in “danger” due to a higher possibility for the dealer to get a good hand than the user. This is considered when the table cards are both the same number and the user has none of that number then there is a possibility the dealer has that number and will very likely win the hand. Thus, the user should fold unless they have a two pair or higher.

This is also considered when the table cards form a two card sequence (ex. 5, 6) and the user does not have a straight as a result of the table cards. There is a heightened possibility that the dealer has a straight. Thus, the user should fold unless they have a two pair or higher.

Also, this model considers if both table cards have the same suit and the user does not have a flush as a result of the table. In this case, there is a heightened possibility that the dealer has a flush. Thus, the user should fold unless they have a two pair or higher.



**Note**: This function is named “playOrFold3” in the code

We calculated its accuracy for 100,000 rounds and the model was correct 73.26% of the time. In conclusion, this model has a similar accuracy compared to the simplest model we developed. However, we believe that this model has the potential to have better accuracy with additional/adjusted constraints because it also considers what cards are being shared with the other player. For example, we found it particularly interesting to investigate how the model reacts when the shared cards create “danger” For example, when the model tells the user to automatically fold when the shared cards create “danger” the models accuracy is 70.27%. Comparatively, when the model only folds due to “danger” if the hand is not a two pair or higher than the models accuracy is 73.26%. Thus, we can conclude that the model's reaction to “danger” from the shared cards can have a significant effect on the model. Additionally, this led to us providing a name for these cases: “false folds”. We define a “false fold” as a hand that is folded even though it has a high chance of winning. The constraints we added are helpful to avoid cases when they fold but shouldn’t have (ex. Has two pairs, flush, etc.). In conclusion, we believe that including the shared cards as a factor and improving the model to reduce “false folds” are key results we discovered that would allow us to develop more accurate models in the future.

1. **Extension to more than two players**

We have considered how we can extend the model to create a game with more than two players. This requires some adjustments to our model in order to solve this problem. We can use the same function that uses the constraints to determine the rank of each hand. These constraints can remain the same but we will call the function additional times to determine the rank for the additional hands. Additionally, we will have to adapt our function that includes the constraints that determine which hand is the winner. This is because our constraints that determine the winner only compare the rank of two different hands and determine the hand that has the higher rank. Thus, if we add additional players we will have to call this function multiple times to compare multiple hands. For example, if we have three players then we can compare the first and second player's hand to determine which hand has the better ranking and then compare the better hand of the first two players and the third player's hand and the higher hand will win. Furthermore, the high card comparator does not need to be extended when we add players. This is because it is built into the function that determines the winner so for each comparison between players, the high card will be compared within the function and used in cases when both players have the same hand rank to determine the winner.

1. **Code implementation**

We broke the code up into many instances for readability and also so that it could function well as a game we needed to split up the encodings.

Different python files handle aspects of the game, the class file is the foundation of our code where the Card class holds the attributes for the number and suits of the cards, the Hand class represents the players hand in the game and the table cards along with the logic to determine hand rankings and the game outcome.

To handle proper dealing of cards we utilized import random to ensure fairness, this simplifies the complexity and makes the code easy to understand and implement.

The centerpiece of our code is the handRanking function, this evaluates the players hands using logical conditions and establishes whether or not it is a certain type of hand (ie. a Straight Flush). We rigorously tested our models ability to determine the rank of different hands. There were instances where the output was not as expected and we had to fix the conditions. We also rigorously tested our determineWinner function. This function was also critical as it accepted two hands and used logical conditions to determine which hand was better.

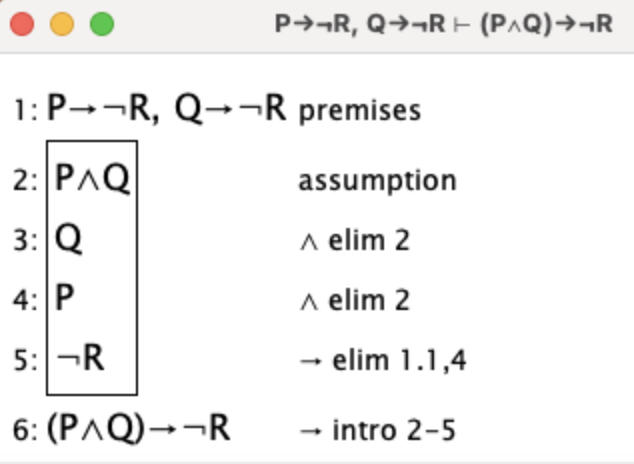
We then have three playOrFold functions which analyze the hands to make a recommendation on whether to play or fold and these decisions vary between them.

The playOrFold functions are used in our function modelAccuracyTester which evaluates their accuracies, using 100 000 simulated poker games to properly recommend to the user whether to fold or play based on the higher percentage.

The game also includes a play again feature where the user can decide if they want to play another round of poker or not. During the gameplay, the program tracks the user’s decision accuracy. We feel this is an interesting feature that will allow users to develop and test their own models/strategies.

**Section 5: JAPE Proofs**

**Sequent 1:** P → ¬R, Q → ¬R |- (P ∧ Q) → ¬R

* ****

**Variables:**

- **P (PlayerHasTwoKings):** Indicates that a player holds two kings.

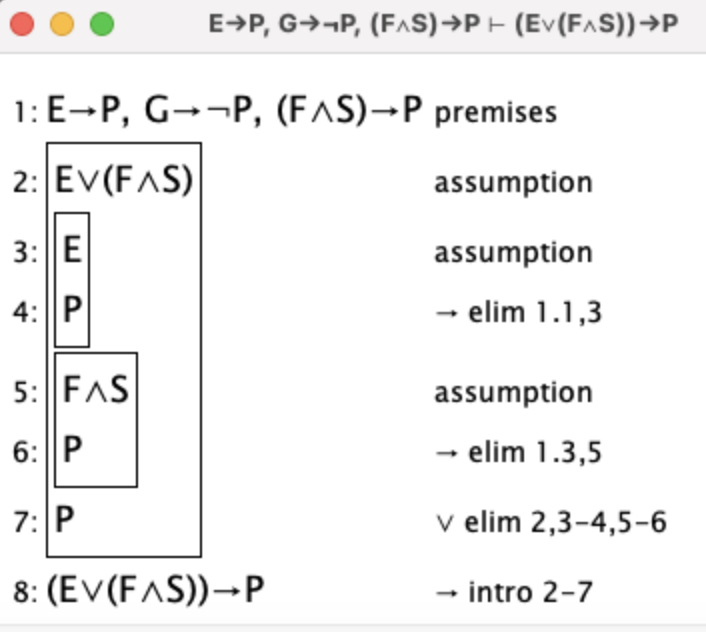
**- Q (TwoKingsOnTable):** Denotes that the other two kings are visible among the community cards.

**- R (OtherPlayerHasKing):** Represents the possibility of another player having a king.

**Description:**

This sequence represents a scenario specific to Three Card and a River Poker. It states that if a player has two kings in their hand and the other two kings are visible on the table, it is logically impossible for any other player to have a king in their hand. This reflects a more realistic and complex situation in the context of the game, adhering to the constraints of Three Card and a River Poker while addressing the feedback for increased complexity.

**Sequent 2:** E → P, G → ¬P, (F ∧ S) → P |- (E ∨ (F ∧ S)) → P

****

**Variables:**

**- E (HighCardAceOrKing):** Represents having a high card that is either an Ace or a King.

**- G (HighCardJackOrLower):** Indicates the high card is a Jack or lower.

**- F (HighCardQueenOrBetter):\*\*** Denotes having a hand with a high card of Queen.

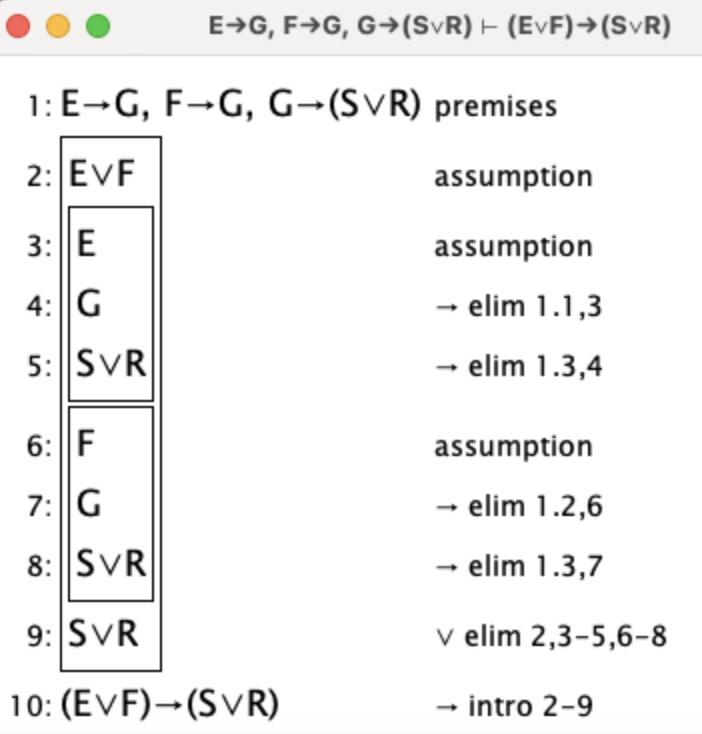
**- P (Play):** Represents the decision to make the Play wager.

**- S (SecondCardSevenOrBetter):** Indicates that the second highest card is a 7 or better.

**Description:**

Captures the strategy for the Play wager decision. A player should play if they have a high card that is an Ace or a King, or a Queen with the second card being seven or better. The player should fold if their high card is a Jack or lower.

**Sequent 3:** E→ G, F → G, G → (S ∨ R) |- (E ∨ F) → (S ∨ R)

****

**Variables:**

**- E (StraightFlush):** Indicates a straight flush.

**- F (ThreeOfAKind):** Signifies three of a kind.

**- G (StrongHand):** Represents having a strong hand.

**- S (WinPot):** Stands for winning the pot.

**- R (OpponentStrongHand):** Possibility of an opponent having a strong hand.

**Description:**

Delineates the implication that possessing either a straight flush or three of a kind leads to a high chance of winning, barring an equally strong opponent hand.

**Section 6: First Order Extension**

In order to extend our project to predicate logic, we can make these following changes to our propositions and constraints:

Firstly, these are the two changes we can make to adjust our propositions:

We can use C(x, y) to model what card the player has. The x represents the suit value which can be selected from the domain {S, C, D, H}. The y represents the number value which can be selected from the domain {2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A}. For example, if S is true and 9 is true, then the card evaluates to true for a 9 of spades. Conversely, if S is true and 7 is true (9 is false), then the card evaluates to false for a 9 of spades.Note: We do not need to change

Also, we can use HU(C1, C2, C3, C4, C5) to model what hand the player has and use HD(C1, C2, C3, C4, C5) to model what hand the dealer has.

We could then use this logic to represent what rank the hand is. For example, P(HU) would evaluate to true if the user's hand contained a pair and false if the user hand did not contain a pair.

Secondly, we can use the changes we made to our propositions to adjust our constraints:

We can apply our propositions to change our hand ranking constraints. For example, we can adjust the constraint that determines whether the hand contains a pair. We could say that:

∀y[(C1 (3) ∧ (C2(3)) ∧ (~C3(3) ∧ C3(y))) → P]

This demonstrates that if the first card is a 3 and the second card is a 3, then for any number value for the third card that is not also a 3 will result in a pair.

We can also adjust our hand constraints to say:

∃x[(C1(x) ∧ C2(H) ∧ C3(H) ∧ C4(H) ∧ C5(H)) → FL]

This demonstrates that if two cards are hearts then there exists a suit (heart) that will result in a flush.

We can also adjust our constraint that compares the hand rankings. For example, we could say:

∀HU[F(HU) → ∃HD(HD ∧ Hu) → WD)]

This demonstrates that for all hands where the user has a flush there exists some hand that the dealer can have that beats this hand and results in a win for the dealer.

Finally, we can adjust our recommendation constraint to be in predicate logic. For example, we could say:

∀HU[~HC(Hu) → RP(Hu)]

This demonstrates that for all hands that are not a high card the model will recommend to play that hand.