Man is a goal seeking animal. His life only has meaning if he is reaching out and striving for his goals.

– Aristotle

University of Alberta

USING COUNTERFACTUAL REGRET MINIMIZATION TO CREATE A COMPETITIVE MULTIPLAYER POKER AGENT

by

Nicholas Abou Risk

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

©Nicholas Abou Risk
Fall 2009
Edmonton, Alberta

Permission is hereby granted to the University of Alberta Libraries to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only. Where the thesis is converted to, or otherwise made available in digital form, the University of Alberta will advise potential users of the thesis of these terms.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Examining Committee

Duane Szafron, Computing Science

Mike Carbonaro, Educational Psychology

Rob Holte, Computing Science

To my family:

my wife Melissa,

my parents Susan and George,

my siblings Joey, Katrina, and Melissa

Abstract

Games have been used to evaluate and advance techniques in the field of Artificial Intelligence since before computers were invented. Many of these games have been deterministic perfect information games (e.g. Chess and Checkers). A deterministic game has no chance element and in a perfect information game, all information is visible to all players. However, many real-world scenarios involving competing agents can be more accurately modeled as stochastic (non-deterministic), imperfect information games, and this dissertation investigates such games. Poker is one such game played by millions of people around the world; it will be used as the testbed of the research presented in this dissertation. For a specific set of games, two-player zero-sum perfect recall games, a recent technique called Counterfactual Regret Minimization (CFR) computes strategies that are provably convergent to an ϵ-Nash equilibrium. A Nash equilibrium strategy is very useful in two-player games as it maximizes its utility against a worst-case opponent. However, once we move to multiplayer games, we lose all theoretical guarantees for CFR. Furthermore, we have no theoretical guarantees about the performance of a strategy from a multiplayer Nash equilibrium against two arbitrary opponents. Despite the lack of theoretical guarantees, my thesis is that CFR-generated agents may perform well in multiplayer games. I created several 3-player limit Texas Hold'em Poker agents and the results of the 2009 Computer Poker Competition demonstrate that these are the strongest 3-player computer Poker agents in the world. I also contend that a good strategy can be obtained by grafting a set of two-player subgame strategies to a 3-player base strategy when one of the players is eliminated.

Acknowledgements

The completion of this dissertation would not have been possible without the help of many important people. Here are some of the people I would like to thank:

- My supervisor, **Duane Szafron**, who I have the utmost respect for: he is a true leader, academic, and genuinely sincere man. Regardless of how many students he was supervising, how many projects he was a part of, or how busy he was, Duane **always** gave me his full attention. I admire him and many of his qualities: leadership, confidence, professionalism, eloquence, clarity, conciseness, among many others. I am very grateful for the motivation and guidance Duane has provided me with throughout this project.

- Programming wizards, **Neil Burch** and **Mike Johanson**, for their patient help and responses to my never-ending programming questions. They definitely made my transition into the Poker Group *much much* smoother. I'm amazed by their programming prowess, they always seemed to instantly conjure up an efficient solution to any problem faced. I especially admire their ability to create code that is both maintainable and extendable.

- The other members of the Computer Poker Research Group for providing ideas, suggestions, feedback, and friendship: **Kevin Waugh**, **John Hawkin**, **Dave Schnizlein**, **Nolan Bard**, **Rich Gibson**, **Morgan Kan**, **Josh Davidson**, **Bryce Paradis**, **Darse Billings**, **Michael Bowling**, **Rob Holte**, and **Jonathan Schaeffer**.

- Friends who have motivated me to live a healthier lifestyle: **Adam Harrison**, **Michael Wadowski**, **John Hawkin**, and **Nolan Bard**.

- My wife, **Melissa Cameron**, my parents, **Susan** and **George Abou Risk**, and the rest of my family, **Joey**, **Katrina**, and **Melissa**, for their love and support. I love you all!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Games have been used to evaluate Artificial Intelligence (AI) algorithms and techniques since before computers were invented. In 1948, Alan Turing began working with a colleague on a chess program which no computer, at the time, was capable of running. Still, in 1952, lacking a viable computer on which to run the program, Turing played out a recorded match by hand (by emulating the computer) but unfortunately lost [13]. Games have some compelling features that make them an excellent testbed for progress in AI research: finite state and action spaces, metrics for evaluating success, and human competition of varying skill levels.

A **perfect information** game corresponds to one in which every player has complete knowledge of the current game state. Examples of perfect information games include Chess, Checkers, Othello, and Backgammon. In each of these two-player games, both players have perfect knowledge of the game state: the positions of all pieces remaining on the board. On the other hand, a game of **imperfect information** corresponds to a game in which some information is hidden from a subset of the players. Examples of imperfect information games include Scrabble and Poker. In Scrabble, each player's tiles are hidden from their opponents. Similarly, players' hole cards are hidden from the other players in Poker.

Every action in a **deterministic** game results in a constant and predictable outcome. For example, attempting to move a checkers piece onto a vacant square along the top-right diagonal always results in the exact same outcome: the piece is successfully moved onto that square. Checkers, Chess, and Othello are all examples of deterministic games. **Stochastic** (or **non-deterministic**) games, however, involve an element of chance. Examples of stochastic games and their source of chance include Backgammon (dice rolls), Scrabble (tile draws), and Poker (card deals).

Within recent years, computer playing programs have been able to compete with and surpass world-class human experts in several popular games:

- **Checkers**. The *Chinook* project was lead by Jonathan Schaeffer at the University of Alberta. In 1994, *Chinook* was declared the Checkers World Champion after the defending Champion, Dr. Marion Tinsley, forfeited due to health concerns [21]. *Chinook* defended its title in 1995. The *Chinook* team announced in 2007, after almost two decades of computations, that they have solved Checkers [20]. Assuming both players play optimally, the game will always end in a draw.

- **Chess**. The *Deep Blue* project was lead by Feng-hsiung Hsu of IBM. In 1997, *Deep Blue* defeated the World Chess Champion Garry Kasparov in a six-game match with two wins, one loss, and three draws [14].

- **Othello**. The *Logistello* program was written by Michael Buro from the University of Alberta. [5]. *Logistello* defeated the Othello World Champion, Takeshi Murakami, 6-0 in a 1997 exhibition match.

- **Scrabble**. The *MAVEN* program was written by Brian Sheppard of IBM. In 1998, a commercial version of *MAVEN* defeated the World Scrabble Champion, Joel Sherman, and runner-up, Matt Graham with a composite score of 6-3 [22].

- **Backgammon**. The *TD-Gammon* program was developed by Gerald Tesauro of IBM. Through self-play and the use of artificial neural nets and temporal difference learning, it has achieved master-level play competing with some of the best players in the world [26].

- **Heads-up Limit Texas Hold'em Poker**. The *Polaris* program was developed by the Computer Poker Research Group (CPRG) at the University of Alberta. In 2008, *Polaris* defeated some of the top Poker players in the world including Matt "Hoss_TBF" Hawrilenko, Bryce "Freedom 25", Nick "stoxtrader" Grudzien, IJay "doughnutz" Palansky, and Kyle "cottonseed" Hendon. The final score for the series of matches was three wins, two losses, and one draw for *Polaris* [7].

While much research and development has been spent on two-player games, very little research efforts have, comparably, been dedicated to multiplayer games. In this dissertation, the term "multiplayer game" refers to a game with three or more players, to differentiate these games from two-player games. However, many games and real-world problems involve interactions between multiple individuals or groups; these interactions may include competition, cooperation, or a combination of both. World-class computer agents (listed above) now exist for several two-player games[1], however, multiplayer computer agents have yet to challenge and defeat a World Champion.

In this dissertation, we present some techniques for generating 3-player limit Texas Hold'em Poker strategies and compare them (by playing many games) against a set of benchmark agents including the 2008 Computer Poker Competition multiplayer champion, *Poki*. We expect that the techniques used in this study can be applied to other multiplayer games and broader real-world domains involving three or more parties.

[1]*MAVEN* was designed for two-player Scrabble games only

1.2 Texas Hold'em Poker

Poker is a class of multiplayer card games encompassing over one hundred variants. Several similarities are shared among the variants: they are played with a standard playing card deck (4 suits of 13 ranks, 52 cards total), there are betting rounds where players can wager chips, and the standard five-card hand ranking system is generally used to determine the winner(s).

Texas Hold'em, often simply shortened to Hold'em, is the most popular Poker variant. It has exploded in popularity within the past decade, due to several factors: the advent of internet Poker rooms where players can play in the privacy of their own homes, the increase in televised Poker, the addition of hole-cameras which allow viewers to see all private cards, and the amateur Chris Moneymaker's unlikely victory in the 2003 World Series of Poker Main Event Championship.

1.2.1 Rankings

Poker hands are made up of exactly five cards and use the hand rankings presented below, from best to worst. An example hand for each ranking is provided in parentheses.

- **Royal Flush.** Ten through ace of the same suit (A♣ K♣ Q♣ J♣ T♣)
- **Straight Flush.** Five cards of consecutive rank and the same suit (5♢ 6♢ 7♢ 8♢ 9♢)
- **Four of a Kind.** Four cards of the same rank (3♡ 3♣ 3♢ 3♠ A♡)
- **Full House.** Three cards of one rank and two cards of another (5♡ 5♣ 5♢ K♠ K♢)
- **Flush.** Five cards of the same suit (A♠ J♠ 6♠ 3♠ 2♠)
- **Straight.** Five cards of consecutive rank (A♣ 2♢ 3♠ 4♣ 5♢)
- **Three of a Kind.** Three cards of the same rank (2♡ 2♣ 2♢ Q♠ J♣)
- **Two Pair.** Two cards of one rank and two cards of another (7♡ 7♣ 3♢ 3♠ 9♢)
- **One Pair.** Two cards of the same rank (4♣ 4♢ A♣ Q♣ 8♠)
- **High Card.** None of the above (K♢ J♡ T♣ 9♢ 2♠)

If several players have the same hand ranking, the winner is determined by card rank. For example, if several players have a *full house*, the player with the highest *three of a kind* rank wins. Similarly, if several players have a *flush*, the player with the highest flush card rank wins.

1.2.2 Rules

The **pot** is the collection of all bets wagered (usually in the form of **chips**) during a game. After each game finishes, a dealer button is moved clockwise by one position around the table. The player sitting in that position is designated the **dealer** or the **button**. The player to the left of the dealer is called the **small blind**. The player to the left of the small blind is called the **big blind**. The small blind and big blind must place mandatory bets (labeled the *small blind* and *big blind*, respectively) into the pot before the game begins. These blinds are used to give players an incentive to play hands.

Before the game begins, the dealer deals two cards, called **hole cards**, face down to each player. For now we ignore the betting rounds. During these rounds, five face-up public cards, called **community cards**, are dealt and shared between all players. After all five community cards have been dealt and all four betting rounds have finished, all remaining players enter a **showdown** where their hands – the best five card combination made from their two hole cards and the five community cards – are revealed and the player with the highest ranking hand wins the pot. If several players have equivalent hands, the pot is split evenly among them.

1.2.3 Actions

Each Texas Hold'em game consists of up to four betting rounds. The possible actions in a betting round include:

- **Bet.** To place a wager into the pot.
- **Raise.** To match the outstanding bet and place an extra bet into the pot. The minimum size of the extra bet is equal to the difference between the two previous bet or raise sizes. For example, if the action on the current betting round was "bet \$2, raise \$5 more to \$7", then the next raise must be at least \$12 total (\$5 higher than \$7).
- **Call.** To match the amount of all outstanding bets.
- **Check.** To pass when there is no outstanding bet.
- **Fold.** To pass and forfeit the current game when there is an outstanding bet. No further actions can be taken by a player who has folded.

1.2.4 Betting Rounds

- **Preflop** is the first betting round where the blinds are placed in the pot and each player receives their hole cards. There are no community cards dealt in this round. The betting begins with the player to the left of the big blind. Postflop refers to all three remaining (i.e. non-preflop) betting rounds.
- The **flop** is the second betting round where three community cards are dealt. The betting begins with the nearest active (i.e. non-folded) player to the dealer's left.
- The **turn** the third betting round where one community card is dealt. The betting begins with the nearest active player to the dealer's left.

- The **river** is the fourth and final betting round where one community card is dealt. The betting begins with the nearest active player to the dealer's left.

1.2.5 Variants

There are several variants of Texas Hold'em Poker. The game often varies in two orthogonal dimensions: the number of players participating and the betting structure.

- The **heads-up** variant includes only two players. Typically, the dealer is the small blind and acts first preflop and last postflop. The **ring** (also called *multiplayer*) variant includes more than two players, usually between three and ten.
- **Limit**, also called *fixed-limit*, is a variant where the amount of each bet and raise is fixed. Each wager is equal to the big blind for the preflop and flop rounds and equal to twice the big blind for the turn and river. For example, when playing a \$10/\$20 limit game the small blind is \$5, the big blind is \$10, bets preflop and on the flop are \$10 and bets on the turn and river are \$20. In this case, \$10 is called a *small bet* and \$20 is called a *big bet*. There is often a **cap** of four bets per player per round in *limit* games. **No-limit** is a variant in which wagers can be any amount between the size of the big blind and all of a player's remaining chips. Betting all remaining chips is called going *all-in*. **Pot-limit** is a similar variant which allows a player to wager any amount between the size of the big blind and the size of the pot. The pot-size includes the amount needed to match any outstanding bets.

1.3 Computer Poker

Each year, the Annual Computer Poker Competition is held (by AAAI or IJCAI) so that academics and amateur programmers alike can determine the relative strength of their Texas Hold'em computer Poker agents. This competition began in 2006 and was designed to drive the development of Artificial Intelligence techniques. In the 2006 AAAI Computer Poker Competition, the only game included was heads-up limit. The next year, heads-up no-limit was added to the competition. Six-player ring limit was added to the 2008 AAAI Computer Poker Competition. This year, in the 2009 IJCAI Computer Poker Competition, the ring limit competition switched from 6-player to 3-player. Each game competition (e.g. heads-up limit) has two sub-competitions where bots are evaluated: *Equilibrium* and *Bankroll*. In the former, agents try not to lose to any other agent; total wins are counted. In the latter, agents attempt to make as much money (and lose as little) as possible over all opponents. This requires some balance between equilibrium and exploitative play.

In the 2008 Computer Poker Competition, the CPRG at the University of Alberta had top competitors in the following games:

- **Heads-up limit**. *Hyperborean08Limit*, an abstract ϵ-Nash equilibrium player, won the limit *Equilibrium* competition in 2008 and placed second to CMU in the *Bankroll* competition. An improved version of this agent, called *Polaris*, was used to defeat world-class human players at the Second Man-Machine Poker Championship, described shortly.

- **Heads-up no-limit**. *Hyperborean08NoLimit*, also an abstract ϵ-Nash equilibrium player, won both the no-limit *Equilibrium* and *Bankroll* competitions in 2008. Designing a competitive agent in no-limit is much harder than in limit since the game state space is enormous due to the number of different bets that can be made. For example, if an agent has a stack size of \$200 and the big blind is \$2, then an agent can make any of the following bets: \$2, \$3, $\ldots$, \$200. Therefore, no-limit requires betting space abstraction in addition to card abstraction. For example, the interpretation of an opponent's possible actions could be translated to a finite set of actions: fold, check/call, pot-sized bet/raise, and all-in. At the present time, the heads-up no-limit agents can still be defeated by strong amateur Poker players.

- **6-player ring limit**. *Poki*, a learning agent designed in 1999 [3], won the 6-player ring limit competition in 2008. There was a total of six entrants, some of which were rule-based (expert knowledge) agents. The fact that a decade-old program won the ring competition is evidence of little to no progress in multiplayer Poker. The size of the betting space for 6-player limit is catastrophically large. Therefore, it was decided to reduce the ring game from 6-player to 3-player for 2009 to reduce the betting space size. The 3-player betting space is over 1500 times larger than that of heads-up and the total game state space is over one million times as large! The hope was that reducing the size of the problem would spur new research in multiplayer Poker. This change also reduced the number of competition hands required to determine statistical significance and the number of relative seating permutations. Despite the fact that *Poki* won the 2008 competition, it still cannot defeat very strong Poker players.

Knowing that the University of Alberta has the top computer Poker players in the world does not suggest how well they would perform against world-class humans. Thus, in 2007, the First Man-Machine Poker Championship was born: a \$50,000 challenge to Poker professionals, Phil "The Unibomber" Laak and Ali Eslami, in a publicized match at AAAI in Vancouver. The championship consisted of four 500-hand **duplicate matches**. In each match, whichever hand was dealt to *Polaris* in "room A" was dealt to a human in "room B". Similarly, whichever hand was dealt to the other human in "room A" was dealt to *Polaris* in "room B". The reason for these duplicate matches is to reduce the element of luck (and thus variance in the results). Although the humans were quite impressed with *Polaris'* play, they still managed to record a victory with 2 wins, 1 draw, and 1 loss.

Still not satisfied, the CPRG improved *Polaris* in several ways: a better card abstraction was created, the equilibrium-finding algorithm was improved in terms of both speed and memory, and the algorithm completed even more iterations than before (thus reducing its exploitability). In 2008, *Polaris* competed against a very strong field of human experts in the Second Man-Machine Poker Championship at the 2008 Gaming Life Expo in Las Vegas. A total of six 500-hand duplicate matches were played and *Polaris* made history by winning the championship (3 wins, 2 losses, and 1 draw).

1.4 Contributions of This Thesis

Counterfactual Regret Minimization (CFR) is a recently developed technique used to find ϵ-Nash equilibria in stochastic, imperfect information games [29]. This technique is very powerful since it requires much less memory than existing techniques: the memory scales with the number of information sets instead of the number of game states [15]. CFR was used to generate the strategy for *Polaris*, the current world champion of heads-up limit Texas Hold'em in both the most recent machine-machine and man-machine competitions.

In this dissertation, we will demonstrate that CFR can be used to generate competitive multiplayer agents which out-perform *Poki*, the existing multiplayer computer Poker champion. Namely, we will show through empirical results that:

- **A *very* abstract multiplayer agent, *Multi2*, generated by CFR out-performs *Poki*.** The 3-player Limit Texas Hold'em game is very large, much larger than heads-up. Even using abstractions, the 3-player game require enormous amounts of computational resources (memory and time) to generate a strategy. Thus, one of the simplest abstractions possible – two hand strength buckets per round (*i.e.*the agent only knows whether its hand is above or below average) – was used to generate a strategy with CFR. Yet this *very* abstract agent, *Multi2*, still out-performs *Poki* in the full game.

- **A master agent comprised of a base player and a team of heads-up experts (HUEs) out-performs both *Poki* and *Multi2*.** As mentioned, the 3-player Limit Texas Hold'em game and associated abstractions are very large. This is mainly due to all of the betting sequences which involve all three players. In practice, however, many hands reduce to a heads-up subgame in the preflop round where one of the players has folded. Examples of such subgames include: the button folds and only the small blind and big blind remain or the button raises and the small blind folds. CFR was used to find ϵ-Nash equilibria for several of these abstract subgames, the resulting agents are called HUEs. The master agent, *Exp_Multi2*, can be viewed as a "coach" that determines when it should substitute in each of its agents. The coach uses a base player for all 3-player scenarios and substitutes in a HUE if their corresponding subgame is reached (based on the betting sequence preflop).

- **Trading off perfect recall of hand strength on previous rounds with more strength buckets per round produces a much stronger 3-player agent.** *Multi2* has only two strength buckets per round but remembers which bucket its hand belonged to for every previous betting round. A new agent, *IR16*, has 16 strength buckets (where higher buckets correspond to stronger hands) but forgets how strong its hand was on all previous betting rounds. With this trade-off, *IR16* out-performs *Exp_Multi2* and the other benchmark agents.

- **Two of the agents created for this research, *IR16* and *Exp_IR16*, are currently the strongest 3-player computer Poker players in the world.** *IR16* placed first in both of the ring limit competitions, *Equilibrium* and *Bankroll*, of the 2009 Computer Poker Competition. *Exp_IR16* (a master agent using *IR16* as the base player) placed second in both of the ring limit competitions.

Chapter 2

Background and Related Work

2.1 Extensive Form Games

An **extensive form game** (or simply *extensive game*) is a representation of a game as a directed tree (often called a **game tree**). The unique initial game state is represented by the root of the tree. Each player's possible decision points are represented by nonterminal nodes called **choice nodes**. The directed edges leaving a choice node represent the legal actions which can be taken by the corresponding player. Terminal nodes represent the end of the game and contain the **payoffs** or **utilities** of all players reaching that point.

The description of an extensive form game, thus far, is sufficient for deterministic games such as Chess and Checkers. However, this description is not sufficient for stochastic games involving chance. For these games, we conveniently represent the chance events (consider the dice rolls in Backgammon or the card deals in Poker) as choice nodes, called **chance nodes**, for a new player called the **chance player**. The directed edges leaving a chance node represent the possible chance outcomes. For example, in a game where the chance player represents the roll of a single die then the possible outcomes include rolling 1, 2, 3, 4, 5, or 6, each with an equal probability of $\frac{1}{6}$.

In imperfect information games, there are often several game states that a player cannot differentiate since the opponents have hidden information (such as their hole cards in Poker). The set of all indistinguishable game states is called an **information set** for that player. Each choice node, in an imperfect information extensive form game, corresponds to the corresponding information set (instead of including a node for every game state). In Poker, there are many more game states than information sets.

2.1.1 Definitions

Definition 1 *[19, p. 200] a finite extensive game with imperfect information has the following components:*

- *A finite set N of **players**.*

- *A finite set H of sequences, the possible* ***histories*** *of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H. $Z \subseteq H$ are the terminal histories (those which are not a prefix of any other sequences). $A(h) = \{a : (h, a) \in H\}$ are the actions available after a nonterminal history $h \in H$,*

- *A function P that assigns to each nonterminal history (each member of $H \backslash Z$) a member of $N \cup \{c\}$. P is the* ***player function****. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$ then chance determines the action taken after history h.*

- *A function f_c that associates with every history h for which $P(h) = c$ a probability measure $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h), where each such probability measure is independent of every other such measure.*

- *For each player $i \in N$ a partition $\mathcal{I}_i$ of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the player $P(h)$ for any $h \in I_i$. $\mathcal{I}_i$ is the* ***information partition*** *of player i; a set $I_i \in \mathcal{I}_i$ is an* ***information set*** *of player i.*

- *For each player $i \in N$ a utility function u_i from the terminal states Z to the reals $\mathbf{R}$. If $N = \{1, 2\}$ and $u_1 = -u_2$, it is a* ***zero-sum extensive game****. Define $\triangle_{u,i} = max_z \, u_i(z) - min_z \, u_i(z)$ to be the range of utilities to player i.*

Definition 2 *A* ***strategy of player i*** *σ_i in an extensive game is a function that assigns a distribution over $A(I_i)$ to each $I_i \in \mathcal{I}_i$, and Σ_i is the set of strategies for player i. A* ***strategy profile*** *σ consists of a strategy for each player, $\sigma_1, \sigma_2, \ldots$, with σ_{-i} referring to all the strategies in σ except σ_i.*

Let $\pi^\sigma(h)$ be the probability of history h occurring if players choose actions according to σ. We can decompose $\pi_i^\sigma(h) = \prod_{i \in N \cup \{c\}} \pi_i^\sigma(h)$ into each player's contribution to this probability. Hence, $\pi_i^\sigma(h)$ is the probability that if player i players according to σ then for all histories h' that are a proper prefix of h with $P(h') = i$, player i takes the corresponding action in h. Let $\pi_{-i}^\sigma(h)$ be the product of all players' contribution (including chance) except player i. For $I \subseteq H$, define $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$, as the probability of reaching a particular information set given σ, with $\pi_i^\sigma(I)$ and $\pi_{-i}^\sigma(I)$ defined similarly.

The overall value to player i of a strategy profile is then the expected payoff of the resulting terminal node, $u_i(\sigma) = \sum_{h \in Z} u_i(h) \, \pi^\sigma(h)$.

2.1.2 Best Response

A **best response** is a strategy which obtains the highest utility possible against the set of all other strategies in a strategy profile, σ. In Poker, this corresponds to the strategy which wins the most money against all of the other players given that their entire strategies have been revealed and remain *static* (i.e. they do not change). An **abstract game best response** is simply a best response to an abstract game given that the best response is confined to playing in the abstract game (as opposed to the real game). Section 2.4 describes a technique which can be used to calculate abstract game best responses. In heads-up games, a best response is sometimes called a *worst-case opponent*.

2.1.3 Nash Equilibrium

Informally, a **Nash equilibrium** is a strategy profile, σ, in which no player can increase their utility by unilaterally changing their strategy. That is, if each player is provided with the static strategies of all the other players and still cannot gain by changing their strategy, then they are all playing a Nash equilibrium. Notice that each player in a Nash equilibrium is a best response to the set of all other players. More formally, a strategy profile for n players, $\sigma = (\sigma_1, \ldots, \sigma_n)$, is a Nash equilibrium if it satisfies the following constraints:

$$\begin{aligned} u_1(\sigma) &\geq \max_{\sigma'_1 \in \Sigma_1} u_1(\sigma'_1, \sigma_2, \ldots, \sigma_n) \\ u_2(\sigma) &\geq \max_{\sigma'_2 \in \Sigma_2} u_2(\sigma_1, \sigma'_2, \ldots, \sigma_n) \\ &\vdots \\ u_n(\sigma) &\geq \max_{\sigma'_n \in \Sigma_n} u_n(\sigma_1, \sigma_2, \ldots, \sigma'_n) \end{aligned} \tag{2.1}$$

Thus, we see that, in a Nash equilibrium, there is no strategy for player i in Σ_i that yields a higher utility against σ_{-i} than its strategy, σ_i, in σ. In fact, in a two-player zero-sum game, each player who plays a Nash equilibrium strategy has the same value regardless of which Nash equilibrium strategy each player plays. In other words, there is no advantage to playing a strategy from one Nash equilibrium over a strategy from a different Nash equilibrium. Unfortunately, however, this is not the case for multiplayer games. The payoffs for each player in one multiplayer Nash equilibrium can differ from the corresponding players' payoffs in another multiplayer Nash equilibrium [18].

Computing an exact Nash equilibrium for a large heads-up game such as Poker (or even an abstraction) is infeasible. Thus, we are often limited to, yet still interested in, finding approximations to Nash equilibria. An **ϵ-Nash equilibrium** is a strategy profile, σ, in which no player can increase their utility by more than ϵ by unilaterally changing their strategy. More formally, a strategy profile for n players is an ϵ-Nash equilibrium if it satisfies the following constraints:

$$\begin{aligned} u_1(\sigma) + \epsilon &\geq \max_{\sigma'_1 \in \Sigma_1} u_1(\sigma'_1, \sigma_2, \ldots, \sigma_n) \\ u_2(\sigma) + \epsilon &\geq \max_{\sigma'_2 \in \Sigma_2} u_2(\sigma_1, \sigma'_2, \ldots, \sigma_n) \\ &\vdots \\ u_n(\sigma) + \epsilon &\geq \max_{\sigma'_n \in \Sigma_n} u_n(\sigma_1, \sigma_2, \ldots, \sigma'_n) \end{aligned} \tag{2.2}$$

Thus, we see that, in an ϵ-Nash equilibrium, there is no strategy for player i in Σ_i that yields a gain of ϵ more utility against σ_{-i} than its strategy, σ_i, in σ. An ϵ-Nash equilibrium is said to be **ϵ-suboptimal** or **exploitable** by ϵ.

2.2 Abstraction

Real-world Poker games are extremely large. In fact, even the smallest full-scale Poker game, heads-up limit Texas Hold'em, is too large to solve[1] with modern technology. Heads-up limit Texas Hold'em has $\binom{52}{2}\binom{50}{2}\binom{48}{3}\binom{45}{1}\binom{44}{1} \approx 5.56 \times 10^{13}$ chance outcomes. Taking into account the betting sequences, this game has about 10^{18} game states. For comparison, the largest unabstracted Poker game solved to date is Rhode Island Hold'em with 10^9 game states. This game was solved by Gilpin and Sandholm [10].

Adding one more player (i.e. 3-player limit Texas Hold'em) results in $\binom{43}{2} = 903$ times as many chance nodes as the heads-up equivalent: $\binom{52}{2}\binom{50}{2}\binom{48}{2}\binom{46}{3}\binom{43}{1}\binom{42}{1} \approx 5.02 \times 10^{16}$. Adding this player also increases the number of betting sequences by a factor of over 1500 resulting in a game with approximately 10^{24} game states. Thus, increasing the number of players from two to three increases the size of the game by a factor of over one million!

Aside from the enormous amount of memory required to store a strategy solution (several petabytes for heads-up limit) [15], solving full-scale Poker games is still intractable. Fortunately though, there are many ways to reduce the size of the game while still maintaining the underlying structure; these reductions are called **abstractions**. Different abstractions vary in the amount that they reduce the size of the game state space and differ in how well solution strategies to the abstract game perform in the full-scale game.

2.2.1 Suit Equivalence Isomorphisms

In Poker, since all four suits have equal strength, we can define an arbitrary ordering of the suits enabling us to map every hand to a canonical hand representation. For example, A♣ K♣ is strategically equivalent to A♡K♡ and J♣ J♡ is strategically equivalent to J♢ J♠. Exploiting these suit symmetries results in no information loss and a reduction of at most $4! = 24$. Due to *self symmetry* (e.g. Q♢ Q♠ is equivalent to Q♠ Q♢, simply reordered), a reduction factor of 24 is not attainable. However, with some cleverly constructed indexing schemes, Gilpin et al. were able to reduce the memory used by a factor of 23.1 [11].

Rank equivalence can also be used to reduce the size of the game with no information loss. For example, with community cards A♡ K♠ Q♣ J♢ T♡, every set of hole cards results in exactly the same 7-card hand ranking (an Ace-high straight).

[1] We use this term loosely to describe finding an ϵ-Nash equilibrium solution for a sufficiently small ϵ

2.2.2 Betting Round Reduction

In most limit Texas Hold'em games, there is a cap of four bets per player per round. One simple and natural abstraction is to reduce the size of the cap. As a consequence, the branching factor (corresponding to the continuing betting sequences) is reduced on every round. For example, reducing the bets per round in limit Texas Hold'em from four to three reduces the branching factor from nine to seven postflop. After experimentally discovering that two bets per round was unsafe, Billings et al. chose an abstraction with three bets per round for their agent, *PsOpti* [2]. They claimed that this reduction did not appear to substantially affect play nor the players' expected values.

In a later study, Zinkevich et al. found an ϵ-Nash equilibrium strategy for an abstract heads-up limit Texas Hold'em game with a two-bet-cap preflop, a three-bet-cap postflop, and bucketing as described below (which reduces card information). The solution was 11 mb/h[2] exploitable by a best response agent in its own abstract game. It was also 27 mb/h exploitable in the abstract game which maintains bucketing but relaxes the betting restriction to the normal four-bet-cap [28].

One caveat regarding multiplayer Poker games is that *betting round reduction* could potentially produce atrocious players. In heads-up, a player has the ability to control the maximum number of bets placed into the pot on each round. More specifically, the opponent can only raise a maximum of one more bet each turn. For example, if a player would like to put in at most three bets on a given round then that player should raise to no more than two bets (allowing the player to call a third bet if the opponent reraises). In multiplayer, however, a player can be *squeezed* (also called *sandwiched* or *whipsawed*) between two or more players who keep raising. This gives players no control over the maximum pot size when playing multi-way pots.

Sometimes an abstract player will fall *off-tree*: the player encounters a situation or decision point which is not defined in its abstract game. In such cases, a default play such as check-and-fold or always-call is often employed.

2.2.3 Bucketing

One of the most effective ways to reduce the size of a Poker game to a more tractable abstract game involves **bucketing** hands. Hands can be partitioned into **buckets** with the intent that hands belonging to the same bucket share strategic similarities. Hands belonging to the same bucket are all played identically in the abstract game.

There are many ways to partition hands into buckets. The CPRG has found that evaluating the "strength" of each hand against a uniformly random distribution of hands is a very effective metric. More formally, we define **hand strength** (HS) as the probability of a hand winning a showdown against a uniformly random hand. The **7 card hand rank** is used as a hand strength metric when there are still cards to be dealt (i.e. the round is not the river) [2]. This metric involves rolling out all

[2] The units used in limit computer Poker are most often small bets per hand (sb/h) or millibets per hand (mb/h). A millibet is 0.001 of a small bet.

undealt community cards and evaluating the probability of a hand winning against a random hand; the metric is called expected hand strength, E[HS], since it is the expectation value of the hand strength. E[HS] is a number in the range [0, 1] since it is an expectation value of probabilities.

A simple example demonstrating hand bucketing is an abstract game with 2 buckets per round. Since we are using E[HS] as our metric, a logical way of bucketing our hands would be to put all of the "above average" hands into the top bucket and all of the "below average" hands into the bottom bucket. That is, we enumerate all possible two-card holdings, sort them based on E[HS] on each round, and then place the strongest half of holdings into the top bucket and the rest into the lower bucket.

Percentile bucketing corresponds to bucketing hands with an approximately equal number of hands in each bucket. As we increase the number of buckets with this bucketing method, we decrease the number of hands in each bucket. However, we are not restricted to placing an equal number of hands in each bucket. Another method involves uniformly partitioning [0, 1] into buckets and is called **uniform bucketing**. For example, with N buckets, we could place all hands with E[HS] in the [0, 1/N) range together, all hands with E[HS] in the [1/N, 2/N) range together, and so on.

In Poker, there are many hands that are not very strong on a given round but have the **potential** of making a very strong hand on a future round (e.g. a straight or flush). It turns out that a very effective alternative metric to E[HS] which intrinsically incorporates potential is called **expected hand strength squared**, $E[HS^2]$ [15]. This metric, which is also in [0, 1], involves taking the expected value of the square of the hand strength. The CPRG has used this metric with great success: it is used in the abstractions of their strongest computer Poker agents. The new agents created for and described in this dissertation used percentile bucketing of expected hand strength squared.

A **bucket sequence** corresponds to the sequence of buckets a hand belonged to on each round. Because Poker involves chance events (i.e. deals between rounds), the strength of a hand can vary from one round to another. **History bucketing**, currently used by the CPRG, is a method of bucketing hands based on their bucket sequences. The number of history buckets on each round grows exponentially: with a N-bucket-per-round abstraction, we have N history buckets preflop, N^2 history buckets on the flop, N^3 history buckets on the turn, and N^4 history buckets on the river. This bucketing method is quite effective as the distribution of possible holdings can be deductively narrowed down based on the transitions along the bucket sequence.

2.2.4 Recall

In **perfect recall** games, there is an assumption that no previously acquired information is forgotten by the players. Thus, in Poker, a player with perfect recall will not forget private cards, public cards, position, or the betting sequence (history of actions). In **imperfect recall** games, the assumption that no previously acquired information is forgotten does not hold. For instance, players can forget their hole cards, actions they have taken (or those taken by their opponents), and so on.

In theory, the guarantees associated with the techniques used by the CPRG to solve abstract Poker games apply only to perfect recall. In practice, however, the strongest agents created by the group have been produced for abstract imperfect recall games (where some history buckets from previous rounds are forgotten). Some of the agents used in this research involve perfect recall (with full history bucketing) and imperfect recall (with incomplete history bucketing).

2.3 Strategy-Knitting

Billings et al. discuss several abstraction techniques involving reducing the number of betting rounds [2]. For instance, an abstraction to the 4-round game could be a 3-round *preflop model* where the river is truncated. In this example, there are several ways in which we could truncate the round: the pot could be awarded to the player with the best hand on the turn, the pot could be distributed amongst the players assuming there is no more betting (the remaining players simply check on the river), or the pot could be distributed amongst the players assuming they put in one bet each on the river. The last two methods described for distributing the pot between players involve computing the expected value of the hands based on a *roll-out* of all possible river cards.

A similar technique to truncating rounds involves having a postflop model where the earlier rounds are *bypassed* and a solution is generated starting at one of the postflop rounds. For example, a 3-round postflop model bypasses the preflop round and starts on the flop. Billings et al. also present the natural idea of having *independent betting rounds* as an abstraction but discuss how this turns out not to be a good assumption in Poker due to the importance of the information associated with the path that lead to that round (i.e. the betting sequence).

Billings et al. were forced to use preflop and postflop models, at the time, due to computational resources since even a bucketed 4-round game was too large. Fortunately, a new technique called Counterfactual Regret Minimization, described in the next section, provides a solution to this bottleneck. However, studying Billings et al.'s previous approaches to *knitting* strategies together might provide us with some insight into solving heads-up subgames of the 3-player game. This approach of using **heads-up experts** was considered due to the enormous size of abstract 3-player games and is described in Chapter 4.

The simplest approach to *knitting* strategies together involves *appending* a postflop model onto a preflop model wherein the agent plays according to the preflop model until it reaches the flop and then uses the postflop model. *PsOpti0* used an *always-call* policy preflop and an appended 3-round postflop model assuming uniform distributions postflop. *PsOpti1* used a 1-round preflop model and an appended postflop model solved for a given pot-size – there are four pot-sizes that reach the flop corresponding to the number of raises – and assuming a uniform distribution of hands for both players. Finally, *PsOpti2* used the distributions on the flop from a 3-round preflop model as input to seven 3-round postflop models (corresponding to the seven preflop betting sequences reaching the flop). As shown by experimental data, *PsOpti1* out-performed *PsOpti2* which out-performed

PsOpti0. However, the authors point out several bugs in *PsOpti2* were discovered after the data was collected. The authors claim that a bugless *PsOpti2* should result in the strongest player among the three presented.

In any case, knitting independent strategies together can cause problems. In Chapter 4, we strategy-knit in the 3-player game and must face these issues.

2.4 Counterfactual Regret Minimization

Regret, also called *opportunity cost*, is the difference between the highest utility attainable (over all actions) and the utility of the action that was actually taken. **Average overall regret** is defined as

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} (u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t)) \tag{2.3}$$

where T is the number of games played and $\sigma_i^t \in \Sigma_i$ is player i's strategy on game t [29].

Counterfactual regret minimization (CFR) is an iterative algorithm that minimizes a form of regret, positive **immediate counterfactual regret**, at each information set [29]. In this paper, Zinkevich et al. show that minimizing positive immediate counterfactual regret, in turn, minimizes the average overall regret. In a two-player zero-sum game, minimizing both players' average overall regret leads to an ϵ-Nash equilibrium strategy profile [29]. Details on the implementation of CFR for general two-player zero-sum perfect recall games and for heads-up Poker can be found in [29] and [15].

CFR was a breakthrough for computing ϵ-Nash equilibrium strategy profiles for imperfect information games since the memory scales with the number of information sets instead of the number of game states, as happens with sequence form solvers [16]. Due to these savings, Zinkevich et al. were able to solve abstract Poker games two orders of magnitude larger than previously solved [29]. Furthermore, another added benefit to using CFR is its ability to parallelize a computation by dividing the game tree into a set of subtrees (e.g. subtrees beginning on the flop, divided based on their unique preflop betting sequences). Another very useful property of CFR is that it can compute an abstract game best responses against a static opponent [15]. CFR can also compute an abstract game best response against multiple static opponents.

While the theoretical guarantees of CFR are somewhat limited (to two-player zero-sum perfect recall games), experiments conducted by the CPRG have demonstrated that CFR seems to be quite robust in generating strong strategy profiles in two-player games with no guarantees. There are no theoretical guarantees for two-player zero-sum *imperfect recall* games. However, the CPRG's strongest heads-up limit Texas Hold'em agents are created with imperfect recall (where only partial information about previous round buckets is remembered). There are also no theoretical guarantees for two-player *nonzero-sum* perfect recall games. However, the CPRG has created very strong heads-up limit Texas Hold'em agents for nonzero-sum games where the agent obtains bonuses for winning hands. These agents have been found to have little exploitability and to actually perform better against human players (due to the agents' tendency to be aggressive and human players' tendency to fold too often). For example, the agents that played in both of the Man-Machine competitions were computed with CFR for nonzero-sum abstract games [7].

CFR also has no theoretical guarantees for multiplayer zero-sum games and, before this study, no research had been conducted on applying CFR to large multiplayer games. As will be shown in later chapters, CFR is also capable of generating world-class computer Poker agents for multiplayer games. Specifically, we generated a 3-player perfect recall agent and a 3-player imperfect recall agent that performed well.

2.5 Agent Evaluation

When new agents are created, we would like to evaluate them so as to provide us with some insight into the effectiveness of the techniques used for their creation. There are several ways to evaluate the performance of agents in Poker games and we will examine some of them here.

2.5.1 Best Response Agents

Best responses provide a metric of the exploitability of an agent. The more exploitable a Poker agent is, the more money a best response strategy will win against it. Abstract game best responses can be computed fairly easily with CFR for heads-up limit Texas Hold'em. However, abstract game best responses for multiplayer Poker games are not as easy to compute due to their enormous size. Generating a best response strategy in CFR takes N CFR runs for an N-player game. Performing just one CFR run (for a relatively low number of iterations) on a 3-player game takes on the order of months. Therefore, we need another method of evaluating our new agents.

2.5.2 Benchmark Agents

To evaluate an agent against other agents, thousands or even millions of hands can be played between them with their corresponding **winrates** (winnings/losses over hands) recorded. There are plenty of heads-up limit Texas Hold'em agents (of varying skill level) which can be used as benchmarks against which to compare new agents. However, relatively little research has been dedicated to multiplayer Poker games and, thus, there are very few benchmark agents.

Poki is an artificially intelligent agent produced by the CPRG and it is capable of playing heads-up and multiplayer games [1]. In fact, *Poki* is the reigning champion of the 2008 Computer Poker Competition in multiplayer limit. *Poki* uses a collection of technologies including an expert system preflop, a formula for mapping hand strength and potential to actions postflop, and Monte Carlo simulations designed to estimate the most profitable action to take against an opponent model.

A **chump** is an agent that disregards its cards and simply takes an action based on the same distribution at every decision point. We can represent a player's distribution over actions as a **prob-**

ability triple, (f, c, r), where f is the probability of folding, c is the probability of checking or calling, r is the probability of betting or raising, and where $f + c + r = 1$. Since *Poki* is the only benchmark agent we have available for multiplayer, we will also use chumps to evaluate our newly created agents. Hopefully we can avoid the embarrassing result of losing to a player that does not even look at its cards!

An **Always-Fold** chump has a probability triple of $(1, 0, 0)$ and always folds on its first action. An **Always-Call** chump has a probability triple of $(0, 1, 0)$ and always checks if there is no outstanding bet and calls otherwise. An **Always-Raise** chump has a probability triple of $(0, 0, 1)$ and always bets if there is no outstanding bet and raises otherwise. The final chump we will use for evaluating our 3-player agents is called **Probe** and it has a probability triple of $(0, 0.5, 0.5)$ so it chooses equally between check/call and bet/raise at every action.

2.5.3 Variance Reduction

Duplicate Bridge is a method of playing tournament Bridge in which the same hands are dealt on two different tables. This method is used to reduce the amount of luck by comparing the scores obtained by the teams receiving the same hands. An analogous approach, called **duplicate matches**, has also been applied to Poker.

In heads-up Poker, duplicate matches involves pre-generating all cards (based on a given seed) that will be dealt until and including the river. Player 1 receives hand A and player 2 receives hand B. After the hand is played out, the players switch positions, their memories are erased (assuming they are artificially intelligent agents), and then player 1 receives hand B and player 2 receives hand A. In addition, the community cards are the same as the previous hand.

By keeping track of the sum of the winnings on both hands for each player, the variance of the outcome of a match of duplicate hands will be lower than simply playing the same number of non-duplicate hands. This is because both players are placed in the exact same situations and their plays are compared to their opponent's plays. Note that the variance will not be reduced completely as the two players will not necessarily end each hand on the same round. For example, one player might choose to fold a very poor hand on the flop based on a probability triple $(0.5, 0.5, 0)$, while the opponent sees the turn card in the same situation with the same probability triple. If the turn is particularly lucky then the player who made the second play is rewarded (even though the strategies at that information set are equal since they had the same probability triple).

In heads-up limit Texas Hold'em, using duplicate matches provides a reduction in the standard deviation per hand by almost four-fold: a typical standard deviation is about ± 6 sb/h [1, p. 13] while it has been empirically measured at ± 1.6 sb/h using duplicate matches [1, p. 17].

Duplicate Poker does not only apply to heads-up; we can generalize it to multiplayer games. Specifically, with N players, we require $N!$ positional permutations so that all positions and relative positions are experienced by all of the players (where the cards are pre-generated and fixed by position). For example, a 3-player game with players A, B, and C has $3! = 6$ positional permutations: ABC, BCA, CAB, ACB, CBA, BAC. In the labeled permutations, the first character arbitrarily represents the position of the button, the second character represents the position of the small blind, and the third and final character represents the position of the big blind. The winnings are averaged over all permutations to obtain a total winrate for the pre-generated sequence of cards.

Chapter 3

The 3-Player Game

3.1 Problems with Multiplayer Games

Two player zero-sum games have convenient theoretical guarantees. For instance, all Nash equilibria have exactly the same value, a strategy from a Nash equilibrium strategy profile is guaranteed to not lose and break even in the worst case scenario (for games with symmetry like Poker where the positions alternate). Unfortunately, these theoretical guarantees do not apply to multiplayer.

3.1.1 Performance of an Equilibrium Player in a Multiplayer Game

In a two player zero sum game, if one player plays a strategy from a Nash equilibrium strategy profile, there is no strategy that the other player can select that reduces the first player's utility. However, in a multiplayer game, if one player plays a strategy from a Nash equilibrium strategy profile, the other players may play strategies that reduce the first player's utility. Consider the following simple game:

- Three players ante 1 unit each
- Each player has a penny and privately chooses heads or tails
- After all three players have chosen a side, they reveal their penny at showdown
- If all three players have chosen the same side, they take back their initial ante
- If exactly two players have chosen the same side, they win and take back their initial ante and split the loser's ante

One of the Nash equilibrium strategy profiles for this modified *matching-pennies* game involves all three players choosing randomly between heads and tails. Each player in this game acts simultaneously and antes the same amount so, with no loss of generality, we can demonstrate that all players playing uniformly random is an equilibrium by fixing two of the players and showing that the other player cannot gain by unilaterally changing its strategy.

Let us represent player i's strategy as $\sigma_i = (P(H), P(T))$ where $P(H)$ is the probability of choosing heads, $P(T)$ is the probability of choosing tails and where $P(H) + P(T) = 1$. Let us arbitrarily fix $\sigma_1 = \sigma_2 = (\frac{1}{2}, \frac{1}{2})$ and let $\sigma_3 = (p, 1-p)$. Regardless of what player 3 chooses, the possible outcomes from players 1 and 2 are $\{$HH, TT, HT, TH$\}$ each with an equal probability of $\frac{1}{4}$. If player 3 chooses heads, it has an expected value of 0 for HH, -1 for TT, +0.5 for HT, and +0.5 for TH. The expected value of player 3 choosing heads is

$$\begin{aligned} EV(H) &= \frac{1}{4}(0) + \frac{1}{4}(-1) + \frac{1}{4}(+0.5) + \frac{1}{4}(+0.5) \\ &= 0 \end{aligned}$$

Similarly, the expected value of player 3 choosing tails is

$$\begin{aligned} EV(T) &= \frac{1}{4}(-1) + \frac{1}{4}(0) + \frac{1}{4}(+0.5) + \frac{1}{4}(+0.5) \\ &= 0 \end{aligned}$$

Given that player 3 chooses heads with probability p and tails with probability $1-p$, a value of

$$\begin{aligned} EV &= p \times EV(H) + (1-p) \times EV(T) \\ &= p(0) + (1-p)(0) \\ &= 0 \end{aligned}$$

is obtained. Thus, we see that, regardless of the strategy chosen by player 3, the same value is obtained. More specifically, deviating from $\sigma_3 = (\frac{1}{2}, \frac{1}{2})$ does not unilaterally improve player 3's utility (or expected value).

However, consider the situation where player 3 chooses the uniformly random strategy, $\sigma_3 = (\frac{1}{2}, \frac{1}{2})$, and both opponents (players 1 and 2) play the static strategies of *always-heads*, $\sigma_1 = \sigma_2 = (1, 0)$. In this case, there are only two possible outcomes, HHH and HHT, with equal probabilities of $\frac{1}{2}$. When player 3 chooses heads (i.e. all three players play heads), player 3 does not gain anything (all three players take back their antes). On the other hand, one ante is lost by player 3 when it chooses tails. Therefore, player 3's value is given by

$$\begin{aligned} EV &= \frac{1}{2}(0) + \frac{1}{2}(-1) \\ &= -\frac{1}{2} \end{aligned}$$

while the value of always choosing heads is 0. It is easy to see that all three players playing *always-heads* is also a Nash equilibrium. From this simple example, we can see that playing the strategy $(\frac{1}{2}, \frac{1}{2})$ which is part of a Nash equilibrium strategy profile can be a disastrous choice if *both* of the opponents change their strategies, regardless of their intentions.

3.1.2 Collusion

In multiplayer Poker games, it is possible for some of the players to form partnerships or alliances with the intent of increasing their combined utility. For example, imagine two players in a 3-player game who decide that they will evenly split their combined winnings after the game has finished. The two players in this situation, said to be in **collusion**, can greatly increase their utility – and, in turn, decrease the utility of their single opponent – by *squeezing* the opponent between them when one of the colluding players has a strong hand.

Collusion can also involve players sharing private information: imagine two players playing online Poker at the same table while also physically being in the same location informing each other of their hole cards. Both examples involve **explicit collusion** where players have made explicit agreements and such collusion is forbidden (i.e. considered illegal, cheating, and/or against terms of service). However, there is a common form of accepted collusion is called **implicit collusion**. For example, in a Poker tournament wherein several players call an all-in from a player with few chips and check until showdown, they increase the probability of the short stack's elimination (and thus increase all remaining players' tournament utilities).

None of the agents created for this research project explicitly use collusion. It is also assumed that none of the participants in the Annual Computer Poker Competition employ any form of explicit collusion (if players are suspected of foul-play then their play will be reviewed and they will potentially be disqualified). While collusion and its consequences is an interesting topic, it has not been studied in this research.

3.2 Other Multiplayer Game Research

Techniques, involving extensions of fictitious play, have already been used to compute ϵ-Nash equilibria for a 3-player Poker game in [8] and [9]. However, the techniques require the ability to efficiently compute best responses. Ganzfried and Sandholm were able to find jam/fold ϵ-Nash equilibria for 3-player no-limit Texas Hold'em. A jam/fold game is a one-round Poker game where each player has two options: fold or bet all remaining chips (*jam* or *go all-in*). Due to the reduced action and betting space, the 3-player jam/fold game is fairly small and best responses can be computed efficiently. Unfortunately, even an abstracted version of the 3-player limit Hold'em game is too large to use Ganzfried and Sandholm's algorithms.

Buro et al. claim to have created an expert-level Skat computer player, *Kermit* (although they admit that the results are not statistically significant) [6]. Skat is a 3-player trick-based card game where two partners play against the other player. *Kermit* uses several techniques including state evaluation, inference, and search. Skat can be fairly safely abstracted as a two-player game due to the partnership between two of the players.

Sturtevant generalized several two-player search algorithms to multiplayer [25]. He generalized two-player minimax to its analog, max^n, for multiplayer. He also presented an algorithm called $paranoid$ where each player assumes that all other players are ganging up against them; this corresponds to a worst-case scenario. These algorithms were used to create state-of-the-art programs in Hearts and Chinese Checkers. However, Sturtevant states that the issues of opponent modeling and handling imperfect information in Poker "dominate over the more traditional search techniques presented" [25, p. 25].

3.3 CFR Results for Multiplayer Toy Games

Running CFR on abstract Poker games for sufficiently long[1] often takes on the order of days to weeks. Thus, smaller games are often necessary because computational game theorists like to test and evaluate new algorithms or methods relatively quickly. In Poker, we still require that these smaller games maintain some of the underlying properties of larger Poker games (such as stochasticity, hidden information, and utility based on actions). The two most common simplified heads-up Poker games are Kuhn and Leduc Hold'em. We will generalize these heads-up games to 3-player and determine whether CFR might be a viable option for generating robust multiplayer strategy profiles. There is no theoretical guarantee that CFR will generate a multiplayer ϵ-Nash equilibrium strategy profile. As illustrated by the *matching-pennies* game in the previous section, even if CFR could compute a multiplayer equilibrium strategy profile, there is no guarantee that any of the strategies would be robust against arbitrary opponents.

3.3.1 3-Player Kuhn

A very simple one-round one-card Poker game was invented in 1950 by Kuhn [17] so that the optimal solution, computed by hand, could be studied. Here, we have extended the definition of Kuhn to three players:

- Each player antes 1 unit each before the cards are dealt.
- Each player is dealt one private card.
- The deck consists of 4 cards (T, J, Q, K) with ranking K $>$ Q $>$ J $>$ T.
- There are no split pots (ties) due to the strict ranking above.
- There is only one betting round with a 1-bet cap.
- When there is no outstanding bet, a player can check or bet 1 unit.
- When there is an outstanding bet, a player can fold or call.

[1]"Sufficiently long" has been determined empirically in the past by obtaining a small enough ϵ for the abstract game that is ϵ-Nash solved using CFR.

The betting tree for 3-player Kuhn is depicted graphically in Figure 3.1. The edges denote the actions (k = check, f = fold, b = bet, c = call) taken by the player given at the left of the figure (P1 is player 1, P2 is player 2, P3 is player 3). The nodes display the number of bets currently in the pot (the initial pot of 3 units is due to the antes). The terminal nodes are coloured based on the number of active players after the betting finishes. Blue represents a player winning without showdown (when the other two players fold) and always has a pot size of 4 units. Green represents one player folding to a bet and the other two entering a showdown and always has a pot size of 5 units. Finally, red represents all three players entering a showdown which always results in a pot size of 6 units (unless all three players check in which case the pot size is 3 units).

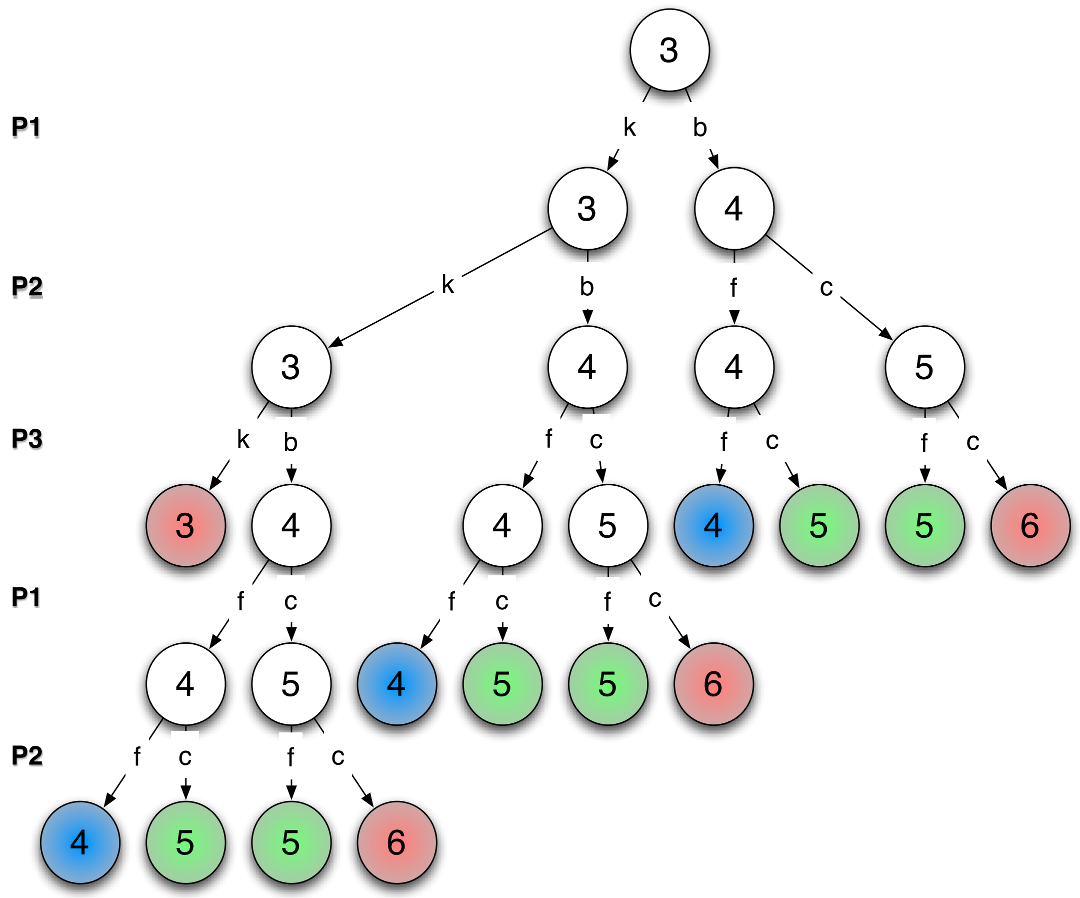


Figure 3.1: The 3-player Kuhn betting tree. Edges mark the actions and nodes display the number of units in the pot. Details are provided in Section 3.3.1.

3.3.2 3-Player Leduc Hold'em

Kuhn intrinsically includes many important aspects of Poker such as bluffing, inducing bluffs, and value betting. However, many important aspects such as multi-round play, community cards, rounds with differing bet sizes, split pots, and the ability to raise are still missing. A small heads-up game, called Leduc Hold'em, which includes all of these important properties was invented by the CPRG and is described in [24]. Here, we have extended the definition of Leduc[2] to three players:

- Each player antes 1 unit before the cards are dealt.
- Each player is dealt one private card.
- The deck consists of 4 ranks and 2 suits (Td, Tc, Jd, Jc, Qd, Qc, Kd, Kc) with ranking K $>$ Q $>$ J $>$ T. Note that suits are arbitrary and do not affect the strength of the hand.
- There can be split pots if two players individually hold cards of the same rank and their hand is stronger than the third player's hand at showdown.
- There are two betting rounds with a 2-bet cap each.
- Each of the betting rounds (let us call the first *preflop* and the second the *flop*) have different bet sizes: preflop bets are 2 units and flop bets are 4 units.
- One community card is dealt before the flop betting.
- Once the flop betting has finished, all active players enter a showdown and the best hand wins the pot (or is split between two winners, as described above). Players either pair their private card or remain unpaired with a high card. A paired hand beats an unpaired hand and flushes and straights do not exist.

3.3.3 Evaluation of 3-Player CFR-Generated Strategies

Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ be the N-player strategy profile generated by running the CFR algorithm for sufficiently long. Recall from Definition 2 that σ_{-i} is the set of all strategies in σ except σ_i. That is, for a 3-player game, we have

$$\begin{aligned} \sigma_{-1} &= \{\sigma_2, \sigma_3\} \\ \sigma_{-2} &= \{\sigma_1, \sigma_3\} \\ \sigma_{-3} &= \{\sigma_1, \sigma_2\} \end{aligned} \tag{3.1}$$

where player 1 acts first, player 2 acts second, player 3 acts third, and the actions cycle in the same order.

If σ is a strategy profile of an ϵ-Nash equilibrium, then no player, i, can gain more than ϵ by deviating from their strategy, σ_i, while σ_{-i} remains fixed. By fixing every strategy in σ_{-i}, CFR can

[2]Leduc Hold'em is henceforth called Leduc for brevity.

generate a best response, σ_i^{BR}, to σ_{-i} from player i's position. Doing so for every player position results in a best response, $\sigma^{BR} = (\sigma_1^{BR}, \sigma_2^{BR}, \sigma_3^{BR})$, to the CFR strategy profile, σ.

The following method can be employed to examine whether a strategy profile generated by CFR (for these 3-player toy games) is an ϵ-Nash equilibrium:

1. Generate a strategy profile $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ by running CFR for 10^8 iterations
2. Compute a best response strategy, σ_1^{BR}, to σ_{-1} using CFR for 10^8 iterations
3. Compute a best response strategy, σ_2^{BR}, to σ_{-2} using CFR for 10^8 iterations
4. Compute a best response strategy, σ_3^{BR}, to σ_{-3} using CFR for 10^8 iterations
5. Combine the three computed best responses into a best response strategy profile, $\sigma^{BR} = (\sigma_1^{BR}, \sigma_2^{BR}, \sigma_3^{BR})$
6. Compute the expected values in each position with three σ's playing against each other
7. Compute the expected values of the best response in each position with one σ^{BR} playing against two σ's
8. Compare σ^{BR}'s expected value in each position to those obtained by σ to determine how much extra σ^{BR} wins
9. If the best response, σ^{BR}, cannot improve by more than ϵ when averaged over all positions by deviating from σ, then σ is an ϵ-Nash equilibrium

The CPRG has chosen 10^8 iterations as a sufficient number for running CFR on 5-bucket heads-up limit Hold'em games. Since there is no penalty for choosing a very large number of iterations[3] and since each CFR run only takes on the order of several hours for these toy games, we have chosen 10^8 iterations. Unfortunately, for multiplayer games, there is no theoretical guarantee of convergence for CFR. Although in practice CFR derives an ϵ-Nash equilibrium strategy for 3-player Kuhn, we cannot show that CFR derives an ϵ-Nash equilibrium strategy for 3-player Leduc.

	Player 1 EV	Player 2 EV	Player 3 EV
CFR	-0.0544877	-0.0418710	0.0963587
BR	-0.0543373	-0.0414071	0.1008150
Improvement	0.0001504	0.0004639	0.0044563

Table 3.1: Expected value of each position in 3-player Kuhn (in antes/h). The final row shows the improvement obtained by the best response.

Tables 3.1 and 3.2 show the expected values obtained by following the 9-step method described above for 3-player Kuhn and Leduc. A tool which walks the whole game tree and determines the

[3]In fact, for an ϵ-Nash equilibrium, increasing the number of iterations decreases the value of ϵ

	Player 1 EV	Player 2 EV	Player 3 EV
CFR	-0.243451	-0.1960510	0.439502
BR	-0.124749	-0.0656892	0.550788
Improvement	0.118702	0.1303618	0.111286

Table 3.2: Expected value of each position in 3-player Leduc (in antes/h). The final row shows the improvement obtained by the best response.

exact value of each position was used to compute the values. The "CFR" row simply consists of playing three σ strategies against each other. The "BR" row consists of playing the best response, σ^{BR}, against two σ strategies (where the expected value is given for the best response). The final row is the difference between the value obtained by the best response and the CFR strategy.

Notice that in Kuhn, the best response cannot gain very much at all by deviating from the CFR solution and is an ϵ-Nash equilibrium with $\epsilon = 0.0017257667$. This is a positive result since there is currently no theory suggesting that CFR should produce an equilibrium for multiplayer games. However, the hope that this result generalizes to larger games is short-lived as we can see that CFR does not produce an ϵ-Nash equilibrium for Leduc. Is it possible that the CFR Leduc strategy is still quite strong against two arbitrary opponents? Of course! We do not have any 3-player benchmark agents for Leduc to answer this question, but we do have benchmark agents for multiplayer limit Texas Hold'em.

3.4 3-player Limit Hold'em

Recall that we have a set of benchmark agents with which to compare (or play) against. Most of these benchmark agents are *chumps* who do not consider their hand (and its strength) but simply take actions with some predetermined frequency at every possible decision point. Namely, our set of benchmark agents is comprised of

- **Always-Call**. With no outstanding bets, this chump always checks. With outstanding bets, this agent always calls.
- **Probe**. This chump will check or call (depending on whether there is an outstanding bet or not) half of the time and raise the other half.
- **Always-Raise**. This chump bets or raises at every opportunity.
- **Poki**. This agent can play with any [legal] number of players. *Poki* is the reigning multiplayer champion of the 2008 Computer Poker Competition.

We require at least one CFR-generated agent to test against our set of benchmark agents. We have generated two types of strategies with CFR: a perfect recall agent and an imperfect recall agent. Note that for the remainder of this dissertation, it is implied that the abstractions use percentile bucketing of the $E[HS^2]$ metric (calculated against *one* random hand). The reason for this decision and possible improvements are provided in Section 6.1.3.

3.4.1 Perfect Recall CFR Agent

Due to the enormous size of the 3-player limit Hold'em game tree, there are very few options available in terms of hand strength bucketing (with current computational resources). In fact, it takes over 32 GB of RAM to store the game tree with no betting abstractions and only two buckets per round. That is, on every round, the agent can only tell if its hand strength compared to a random hand is above or below average! Intuitively, we believed that this would correspond to an agent whose play is very poor but, with no known alternatives, we decided to try it anyway.

We ran CFR on a 2-bucket perfect recall abstraction for 20 million iterations on a 32 GB RAM machine and called the agent **Multi2**. However, we were required to reduce the betting on the river from a 4-bet cap to a 3-bet cap to fit the abstraction into memory. In the event that *Multi2* falls off-tree while playing the real game (i.e. the betting is capped on the river), *Multi2* will simply call. The 20 million iteration CFR computation took about three weeks to run. Since 100 million (10^8) iterations would require 15 weeks of computation and since we only had limited access to the 32 GB RAM machine, only 20 million iterations were completed. Section 3.5.3 describes our evaluation of *Multi2*.

3.4.2 Imperfect Recall CFR Agent

In perfect recall, the number of bucket sequences increases exponentially with the number of betting rounds. With N buckets per round, we have N bucket sequences preflop, N^2 bucket sequences on the flop, N^3 bucket sequences on the turn, and N^4 bucket sequences on the river. Even with a 2-bucket abstraction, this results in $2^4 = 16$ bucket sequences on the river. Since the point of using abstractions is to reduce the memory requirements to something manageable, using some form of imperfect recall seems like it might be a viable option (to reduce the memory requirements). Specifically, the buckets of all previous rounds are forgotten and only the betting sequence and current betting round's bucket is remembered. There are no guarantees that CFR will converge to an equilibrium with an imperfect recall abstraction in the two player game. However, in practice, the CPRG has found that the strongest heads-up limit Hold'em agents have been created using imperfect recall. With multiplayer games, we have no guarantee that CFR will converge with perfect recall so using imperfect recall does not reduce our non-existent theoretical guarantees.

Due to the exponential growth of the bucket sequences per round, most of the memory used for an imperfect recall limit Hold'em game corresponds to that used on the river. With a 2-bucket perfect recall abstraction, there are 16 bucket sequences on the river corresponding to a game approximately the size of a 16-bucket imperfect recall abstraction. Thus, we decided to create a 16-bucket imperfect recall abstraction (the abstraction is aptly named **IR16**) for the 3-player game to compare the strength of these two similar-sized abstractions.

We ran CFR on this abstraction for 20 million iterations (to correspond to the number of iterations used for *Multi2*) and named the resulting agent **IR16_20M**. We also continued the CFR computation until (and beyond) the 2009 Computer Poker Competition. One of the agents submitted to the 2009 CPC completed 43 million iterations and was unimaginatively named **IR16_43M**.

With access to a new 64 GB RAM machine, we were able to include a 4-bet cap on the river as compared to *Multi2*'s 3-bet cap. The CFR computation took on the order of one month to reach 43 million iterations. Section Section 3.5.3 describes our evaluation of *IR16*.

3.5 Evaluation of CFR-Generated Strategies in 3-Player Limit Hold'em

Although computing best responses in very small games is quite feasible and provides a metric for how much each player could improve by unilaterally deviating from their strategy, doing so for full-scale Poker games is often infeasible. Even if the best response computation was feasible, it would be of limited use as it provides no indication of the strength[4] of an agent against *two arbitrary opponents*. Thus, the method of evaluating the strength of the agents created in this thesis will involve playing millions of Poker hands against benchmark agents as well as the other newly created agents. This method of using tournament play (or *cross-play*) to rank agents is also used in the Annual Computer Poker Competition.

3.5.1 Positional Permutations and Duplicate Matches

Suppose we have a match between three players A, B, and C. To reduce variance, we would like each player to play in the same situation (i.e. same position and cards) as the others. We would also like to exhaust all possible relative position orderings; there are only two of them in 3-player games with unique agents. Hence, we have $3! = 6$ permutations of players for every set of pre-generated cards since there are three positions (button, small blind, big blind) and two relative position orderings:

$$ABC$$
$$BCA$$
$$CAB$$
$$ACB$$
$$CBA$$
$$BAC$$

We can average each player's winrate over all of the above permutations to get a winrate for that particular matchup.

[4]We loosely define a 3-player agent being stronger than another if it has a higher winrate than the other against two arbitrary opponents

3.5.2 Tournament and Competitions

The strength of agents can be evaluated in several different ways. Here, we evaluate the strength of agents in exactly the same way agents are evaluated in the Annual Computer Poker Competition. A 3-player tournament with N agents (or entrants) consists of $\binom{N}{3}$ distinct 3-player matchups where each agent plays against $\binom{N-1}{2}$ pairs of opponents.

- **Bankroll Competition**. In this competition, all players are simply ranked based on their total winrate against all pairs of opponents. The aim of this competition is to determine which agent exploits its opponents by the most.

- **Equilibrium Competition**. Also called the *instant run-off bankroll* competition, this competition is determined by removing the worst-ranked player from the *bankroll* competition and determining a new bankroll ranking assuming the worst-player had not entered. This process continues recursively (always eliminating the worst remaining player) until exactly three players remain in which case they are ranked by winrate in that single matchup. The aim of this competition is to determine which agent is the most robust or least exploitable.

The outcome of both of the above competitions can be determined with exactly the same set of tournament hand data (i.e. it is not necessary to run two separate tournaments). It is also not necessary to run a new competition once a player is eliminated from the *Equilibrium* competition as we can simply compute new winrates after removing all matchups involving the eliminated player.

3.5.3 Experimental Results

A tournament with all of our 3-player limit Hold'em agents

- *Always-Call* (AC)
- *Probe*
- *Always-Raise* (AR)
- *Poki*
- *Multi2*
- *IR16_20M*
- *IR16_43M*

was devised. This tournament consisted of $\binom{7}{3} = 35$ matchups where each player played against $\binom{6}{2} = 15$ pairs of opponents. Each matchup consisted of 1.2 million hands (6 positional permutations of 200,000 hands).

Results are displayed in cross-tables as seen in Figures 3.2 to 3.6 with one player in each row and two players in each column. The winrate of the row player versus the two column players is provided as an entry at their intersection. The winrates of all three players sum to zero (and the sum of all winrate entries in the table is zero). The final column represents the overall winrate of the row player against all pairs of opponents; this is the metric used to evaluate and rank the agents. Note that matchups without three distinct players are greyed out. Positive winrates are highlighted in green and negative winrates are highlighted in red. All winrates are provided in units of small bets per hand (sb/h). For comparison, the *Always-Fold* strategy loses 0.500 sb/h in a 3-player game.

We computed 95% confidence intervals for each matchup. The size of the confidence intervals varied from 0.014 to 0.079 sb/h. However, $\pm$ 0.039 (i.e. interval size of 0.078) only occurred in the matches with *Always-Raise* and *Probe* present. Most of the confidence intervals were within $\pm$ 0.01 sb/h.

From Figure 3.2, we see that the rankings of the agents in the *bankroll* competition are (reading from bottom row up):

1. *Multi2*
2. *IR16_20M*
3. *IR16_43M*
4. *Poki*
5. *Always-Raise*
6. *Probe*
7. *Always-Call*

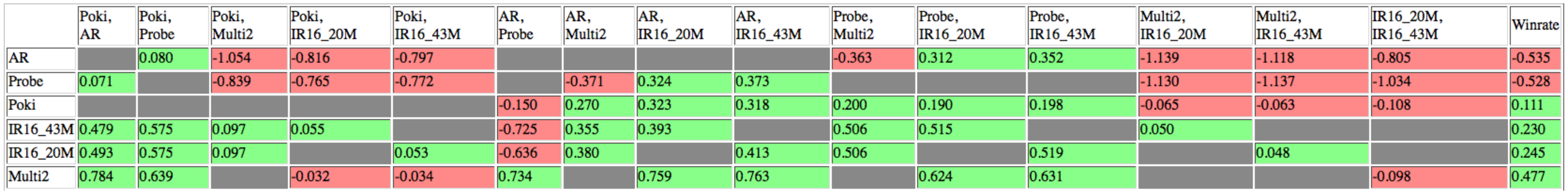
From Figure 3.6, we can determine the rankings of the top three agents in the *Equilibrium* competition. The fourth-ranked agent is the last player to be eliminated and we continue this process until the first agent eliminated is ranked last. Here are the final rankings for the *Equilibrium* competition:

1. *IR16_43M*
2. *IR16_20M*
3. *Multi2*
4. *Poki*
5. *Probe*
6. *Always-Raise*
7. *Always-Call*

Notice that all three of our new CFR-generated agents are ranked top three in both competitions and out-perform the 2008 CPC multiplayer champion (the world's strongest multiplayer computer Poker agent). *Multi2* is clearly the best exploitative player in this tournament and is able to really beat up on chumps. However, when only one chump (*Probe*) remains, *Multi2* is exploited by the *IR16* agents and falls behind the *IR16* agents in the run-off. The *IR16* agents are definitely the least

	Poki, AC	Poki, AR	Poki, Probe	Poki, Multi2	Poki, IR16_20M	Poki, IR16_43M	AC, AR	AC, Probe	AC, Multi2	AC, IR16_20M	AC, IR16_43M	AR, Probe	AR, Multi2	AR, IR16_20M	AR, IR16_43M	Probe, Multi2	Probe, IR16_20M	Probe, IR16_43M	Multi2, IR16_20M	Multi2, IR16_43M	IR16_20M, IR16_43M	Winrate
AC		-0.601	-0.469	-0.972	-1.042	-1.046						-0.004	-0.541	-0.212	-0.223	-0.450	-0.368	-0.367	-0.911	-0.908	-0.992	-0.607
Probe	-0.467	0.071		-0.839	-0.765	-0.772	-0.003		-0.448	-0.369	-0.364		-0.371	0.324	0.373				-1.130	-1.137	-1.034	-0.462
AR	-0.597		0.080	-1.054	-0.816	-0.797		0.006	-0.545	-0.216	-0.218					-0.363	0.312	0.352	-1.139	-1.118	-0.805	-0.461
Poki							1.197	0.936	0.574	0.545	0.540	-0.150	0.270	0.323	0.318	0.200	0.190	0.198	-0.065	-0.063	-0.108	0.327
IR16_43M	0.507	0.479	0.575	0.097	0.055		0.441	0.731	0.516	0.503		-0.725	0.355	0.393		0.506	0.515		0.050			0.333
IR16_20M	0.497	0.493	0.575	0.097		0.053	0.428	0.737	0.510		0.489	-0.636	0.380		0.413	0.506		0.519		0.048		0.341
Multi2	0.398	0.784	0.639		-0.032	-0.034	1.086	0.899		0.401	0.393	0.734		0.759	0.763		0.624	0.631			-0.098	0.530

Figure 3.2: A cross-table showing the results of a 3-player tournament with 7 agents. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (final column) are given within $\pm$ 0.006 sb/h with 95% confidence.

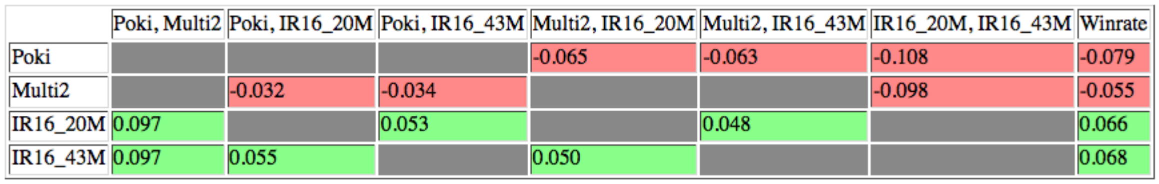


	Poki, AR	Poki, Probe	Poki, Multi2	Poki, IR16_20M	Poki, IR16_43M	AR, Probe	AR, Multi2	AR, IR16_20M	AR, IR16_43M	Probe, Multi2	Probe, IR16_20M	Probe, IR16_43M	Multi2, IR16_20M	Multi2, IR16_43M	IR16_20M, IR16_43M	Winrate
AR		0.080	-1.054	-0.816	-0.797					-0.363	0.312	0.352	-1.139	-1.118	-0.805	-0.535
Probe	0.071		-0.839	-0.765	-0.772		-0.371	0.324	0.373				-1.130	-1.137	-1.034	-0.528
Poki						-0.150	0.270	0.323	0.318	0.200	0.190	0.198	-0.065	-0.063	-0.108	0.111
IR16_43M	0.479	0.575	0.097	0.055		-0.725	0.355	0.393		0.506	0.515		0.050			0.230
IR16_20M	0.493	0.575	0.097		0.053	-0.636	0.380		0.413	0.506		0.519		0.048		0.245
Multi2	0.784	0.639		-0.032	-0.034	0.734		0.759	0.763		0.624	0.631			-0.098	0.477

Figure 3.3: A cross-table showing the results of the remaining 6 agents in a 3-player *Equilibrium* competition with 7 agents after eliminating the worst player. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (final column) are within ± 0.007 sb/h with 95% confidence.

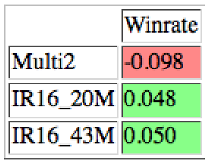
	Poki, Probe	Poki, Multi2	Poki, IR16_20M	Poki, IR16_43M	Probe, Multi2	Probe, IR16_20M	Probe, IR16_43M	Multi2, IR16_20M	Multi2, IR16_43M	IR16_20M, IR16_43M	Winrate
Probe		-0.839	-0.765	-0.772				-1.130	-1.137	-1.034	-0.946
Poki					0.200	0.190	0.198	-0.065	-0.063	-0.108	0.058
Multi2	0.639		-0.032	-0.034		0.624	0.631			-0.098	0.289
IR16_20M	0.575	0.097		0.053	0.506		0.519		0.048		0.300
IR16_43M	0.575	0.097	0.055		0.506	0.515		0.050			0.300

Figure 3.4: A cross-table showing the results of the remaining 5 agents in a 3-player *Equilibrium* competition with 7 agents after eliminating the two worst players. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (final column) are within $\pm$ 0.005 sb/h with 95% confidence.



	Poki, Multi2	Poki, IR16_20M	Poki, IR16_43M	Multi2, IR16_20M	Multi2, IR16_43M	IR16_20M, IR16_43M	Winrate
Poki				-0.065	-0.063	-0.108	-0.079
Multi2		-0.032	-0.034			-0.098	-0.055
IR16_20M	0.097		0.053		0.048		0.066
IR16_43M	0.097	0.055		0.050			0.068

Figure 3.5: A cross-table showing the results of the remaining 4 agents in a 3-player *Equilibrium* competition with 7 agents after eliminating the three worst players. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (final column) are within ± 0.005 sb/h with 95% confidence.



	Winrate
Multi2	-0.098
IR16_20M	0.048
IR16_43M	0.050

Figure 3.6: A cross-table showing the final three agents in a 3-player *Equilibrium* competition with 7 agents after eliminating the worst player. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The winrates are within ± 0.009 sb/h with 95% confidence.

exploited agents in this tournament, only ever losing chips in one matchup (the case where they are sandwiched between *Always-Raise* and *Probe*). *Poki* also loses the most against *Always-Raise* and *Probe*. On the other hand, *Multi2* is the only agent to win in this matchup and does so by a substantial amount.

It turns out that the intuitive assumption that the 2-bucket abstraction plays very poorly was incorrect. It is also worth noting that even after 20 million iterations, a fairly strong 3-player agent can be created (recall that 100 million iterations are typically used for a 5-bucket *heads-up* limit Hold'em abstraction). After about twice as many iterations, there seems to be little gain. Perhaps the agent is trading off exploitative default play with more robust or unexploitable play.

There are several other observations we can make from Table 3.2 (this table contains results from every matchup). Notice that even though all agents in the matchup between *Always-Call*, *Always-Raise*, and *Probe* showdown every hand, the winrates are not zero. This is not a mistake; the reason is that the pot sizes will not be the same over all duplicate matches due to relative positions between the players. *Always-Call* ends up losing more than if it had folded every single hand and does not have a positive winrate in any of the matchups. The most amount won in any matchup was by *Poki* playing against *Always-Call* and *Always-Raise* by over one small bet per hand (also accomplished by *Multi2* but not by as much)!

3.5.4 Self-Play and 3-Player Standard Deviations

A typical standard deviation for heads-up limit Hold'em is about 6 sb/h and is reduced to 1.6 sb/h by using duplicate matches [4]. To determine duplicate match typical standard deviations for 3-

player limit Hold'em games, we ran *self-play* matches consisting of three of the same agents playing against each other. Each of the seven matchups involved 200,000 hands. Since these were duplicate matches, the winrate of each player was exactly zero[5].

Agent	Standard Deviation (sb/h)
Always-Call	1.39
Poki	3.36
IR16_20M	4.65
IR16_43M	4.66
Multi2	4.98
Probe	21.75
Always-Raise	33.32

Table 3.3: The duplicate match standard deviation per hand for matchups between three identical agents (via self-play). These were computed based on 6 permutations of 200,000 hands giving 1.2 million hands total.

Table 3.3 shows some duplicate match standard deviations for our set of seven 3-player agents. The variance for three *Always-Call* agents is quite low since each player is always putting in one small bet preflop (calling the big blind) and then checking until showdown postflop. The variance for *Probe* and *Always-Raise* is much higher since there is so much raising (for instance, *Always-Raise* caps every betting round for a total investment of 24 small bets per hand). Examining the standard deviations of the competent agents (i.e. non-chumps) and some standard deviations from the tournament matchups described in the previous section, a typical duplicate match standard deviation for 3-player limit Hold'em is around 4 to 5 sb/h for competent players. Naturally, however, the standard deviation is higher when several aggressive chumps are playing.

[5]This is only partially true: since Poki has its own random seed, it was not exactly zero but very close. However, the rest of the agents had winrates of *exactly* zero.

Chapter 4

A Master Agent Using Heads-Up Experts

Chapter 3 provided evidence that CFR is capable of generating very strong 3-player limit Hold'em agents even though they used very coarse-grained abstractions. Abstractions using both perfect and imperfect recall were used to create agents stronger than *Poki*, the strongest known multiplayer computer Poker agent in the world. The heads-up limit Hold'em agent, *Polaris*, created by the CPRG now plays at a world-class level. Heads-up situations arise quite frequently in 3-player limit Hold'em when one of the players folds. In fact, the hand will end up heads-up more than *half* of the time in many 3-player games when none of the players are novices or chumps. This chapter aims to exploit the fact that most hands end up heads-up before the flop.

4.1 Self-Play Preflop Folding Frequencies

We know that *Poki* and our CFR-generated agents play very well (at least compared to computer agents). Perhaps by looking at common preflop betting sequences in games between these agents, we can gain some insight into which subtrees of the game are visited most frequently. More specifically, we would like to see how frequently hands end up heads-up before the flop since CFR has nice theoretical guarantees for heads-up zero-sum perfect recall games.

Tables 4.1 to 4.4 show empirical frequencies of different bet sequences with a preflop fold action. A raise is denoted by '*r*', a call denoted by '*c*', and a fold denoted by '*f*'. The frequencies given in each table were computed from 200,000 hands of self-play between three of the same agents. The agents used for self-play include *Poki*, *Multi2*, *IR16_20M*, and *IR16_43M*. The entries have been rounded to two decimal places; the frequencies reading "0.00%" have actually occurred a non-zero yet very infrequent percentage of the time.

Notice that about 50% to 70% of the hands played result in heads-up play before the flop. The most frequent bet sequences with a preflop fold include *f, rf, rrf, rcf, cf, crf*. Creating experts for these common subgame situations should provide great improvements. Before delving into more

Bet Sequence	Frequency (%)
f	20.67
cf	16.43
rf	9.55
crf	3.35
rcf	1.52
rrf	0.54
crcf	0.00
ccrf	0.00
crrf	0.00
Total	52.05

Table 4.1: Empirical frequencies of bet sequences observed with a preflop fold action in 1.2 million hands of self-play between three *Poki* agents.

Bet Sequence	Frequency (%)
f	29.27
rf	25.60
rcf	9.96
rrf	1.51
cf	0.02
crf	0.01
rcrcrf	0.00
Total	66.38

Table 4.2: Empirical frequencies of bet sequences observed with a preflop fold action in 1.2 million hands of self-play between three *Multi2* agents.

details about heads-up subgames, however, let us first define some notational conventions.

4.2 Nomenclature

Notational conventions for representing isomorphic Poker hands have been adopted by most Texas Hold'em Poker literature [23]. These conventions will also be used here. Namely,

- Each of the 13 card ranks are represented by a single character. In increasing order, they are {2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A}.
- Each of the four suits are also represented by a single character. In alphabetical order, they are {c, d, h, s} which represent {♣, ♢, ♡, ♠} respectively.
- Hands fall into one of three categories preflop: pairs, offsuit hands, and suited hands. Pairs correspond to both hole cards having the same rank. Offsuit hands consist of two hole cards of different rank and suit. Finally, suited hands are comprised of two hole cards with the same suit but different rank. A suffix of *'s'* is used to denote suited hands, *'o'* to denote offsuit hands, and no suffix is used for pairs.
- Hole cards are sorted with highest rank appearing first (e.g. 9♢K♠ is represented by K9o).

Bet Sequence	Frequency (%)
f	36.03
rf	26.40
rrf	6.64
rcf	0.66
cf	0.09
crf	0.04
rrrf	0.01
ccrcf	0.00
crrf	0.00
ccrrf	0.00
rcrrf	0.00
rcrf	0.00
Total	69.88

Table 4.3: Empirical frequencies of bet sequences observed with a preflop fold action in 1.2 million hands of self-play between three *IR16_20M* agents.

Bet Sequence	Frequency (%)
f	36.14
rf	26.37
rrf	6.61
rcf	0.65
cf	0.04
crf	0.02
rrrf	0.01
crrf	0.00
ccrrf	0.00
ccrcf	0.00
rcrrf	0.00
rcrf	0.00
Total	69.85

Table 4.4: Empirical frequencies of bet sequences observed with a preflop fold action in 1.2 million hands of self-play between three *IR16_43M* agents.

For example, AKs $\in$ {AcKc, AdKd, AhKh, AsKs} = {A♣K♣, A♢K♢, A♡K♡, A♠K♠}. Similarly, TT $\in$ {TcTd, TcTh, TcTs, TdTh, TdTs, ThTs} and KJo $\in$ {KcJd, KcJh, KcJs, KdJc, KdJh, KdJs, KhJc, KhJd, KhJs, KsJc, KsJd, KsJh}. For conciseness, non-pairs can often be grouped together if they are played in precisely the same way (e.g. JT represents JTs or JTo).

As seen in the above examples, with no loss of generality, there are 6 combinations of each pair, 4 combinations of each suited hand, and 12 combinations of each offsuit hand. There are exactly 169 unique preflop hands: there are 13 pairs (one corresponding to each rank), $\binom{13}{2} = 78$ suited hands, and $\binom{13}{2} = 78$ offsuit hands. By multiplying the number of hand types by their corresponding combinations, we can verify the total number of preflop hands: $13 \times 6 + 78 \times 4 + 78 \times 12 = 1326 = \binom{52}{2}$.

Given that we have now defined some convenient notation, we can easily graphically represent a preflop strategy with a *preflop hand matrix* as seen in Figures 4.2 to 4.4. There are $13 \times 13 = 169$

hands in the table corresponding to every unique preflop hand isomorphism. Pairs appear on the diagonal from the top left to the bottom right. All suited hands appear above the diagonal and all offsuit hands appear below the diagonal. We have arbitrarily chosen to denote raising hands in green, calling hands in yellow, and folding hands in red.

4.3 Heads-Up Experts

As was previously stated and later shown empirically, most 3-player hands end up heads-up before the flop. Since the resulting heads-up subgame is two-player, zero-sum, and perfect recall, we can use CFR and are guaranteed to find an ϵ-Nash equilibrium in the subgame. Moreover, since the heads-up subtree is much smaller (in terms of game states) than the corresponding 3-player tree, strategies using much larger abstractions can be used. We call the agents resulting from the CFR solution to these heads-up subgames **heads-up experts** (HUEs).

Figure 4.1 shows an example of a partial preflop 3-player tree with HUE subtrees coloured in red (the bet sequences leading to these nodes are *f, rf, rrf, rcf, cf, crf*). Actions are labeled along the branches. Nodes contain a value denoting the number of buckets (out of five) that the parent node propagates to that node via the action-labeled edge connecting them.

Figures 4.2 and 4.3 show *Poki*'s preflop strategy on the button depending on the number of hands it plays, tight versus loose. Tight means that it is more conservative or plays less hands while loose means that it is more liberal in the hands that it chooses to play. Notice that the raising hands for both settings are exactly the same and the only way the loose *Poki* changes its distribution is by changing some of the folding hands into calling hands. The *Poki* used in this dissertation refers to the loose version and it was chosen because it out-performed the tight version in tests against chumps and it won the 2008 Computer Poker Competition.

Figure 4.4 shows the preflop strategy suggested for the button by Nick "stoxtrader" Grudzien and Geoff Zobags in their limit Hold'em book [12]. The authors suggest a raise-or-fold strategy where players do not call preflop. Their suggestions involve raising 40% of hands and folding the remaining 60%. If we wished to represent a button strategy similar to this one in a 5-bucket abstraction, we could have the agent raise the 2 strongest buckets (40%) and fold the 3 weakest buckets (60%).

This is the basic idea of creating HUEs: assume some distributions of buckets propagated to a HUE subtree by each of the players. Once the heads-up subtree is reached, solve the corresponding heads-up subgame. Later, substitute this agent or its strategy in place of the 3-player agent when the HUE subtree is reached in-game. Hopefully, if the assumed distributions are reasonable, this method will provide an improvement to the winrate of the 3-player agent. Notice the similarity between HUEs and the method used for *PsOpti* described in Section 2.3.

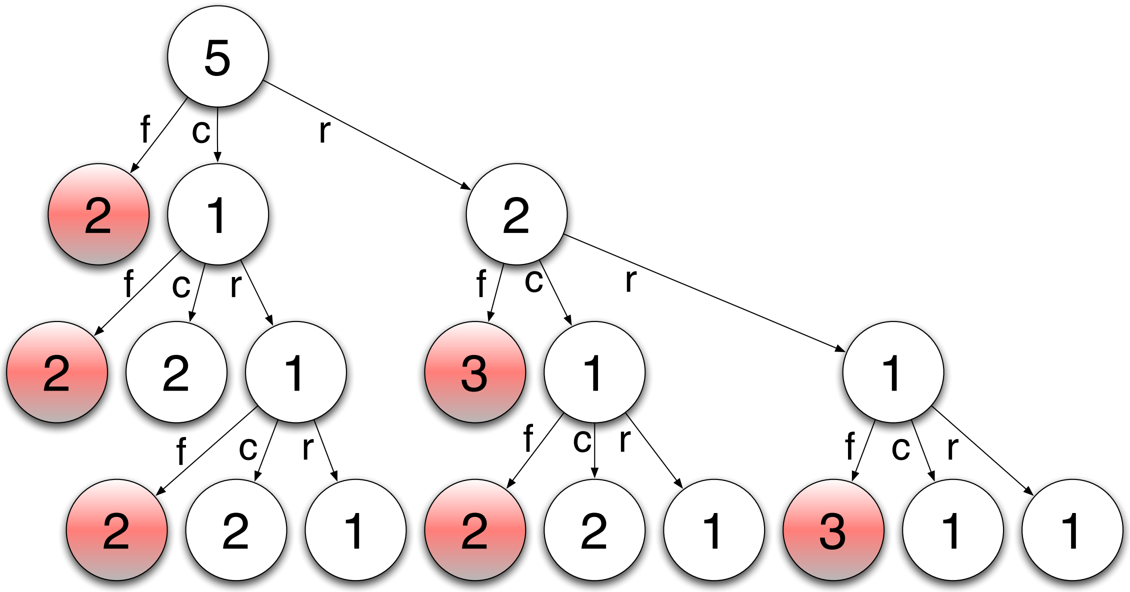


Figure 4.1: Partial representation of a 3-player preflop tree and an example of a possible seeded bucket strategy. Heads-up expert (HUE) subtrees are coloured in red (*f, rf, rrf, rcf, cf, crf*).

🟩 AA	🟩 AKs	🟩 AQs	🟩 AJs	🟩 ATs	🟩 A9s	🟩 A8s	🟩 A7s	🟩 A6s	🟩 A5s	🟩 A4s	🟩 A3s	🟨 A2s
🟩 AKo	🟩 KK	🟩 KQs	🟩 KJs	🟩 KTs	🟩 K9s	🟩 K8s	🟨 K7s	🟨 K6s	🟨 K5s	🟨 K4s	🟨 K3s	🟥 K2s
🟩 AQo	🟩 KQo	🟩 QQ	🟩 QJs	🟩 QTs	🟩 Q9s	🟨 Q8s	🟨 Q7s	🟨 Q6s	🟨 Q5s	🟥 Q4s	🟥 Q3s	🟥 Q2s
🟩 AJo	🟩 KJo	🟩 QJo	🟩 JJ	🟩 JTs	🟩 J9s	🟨 J8s	🟨 J7s	🟥 J6s	🟥 J5s	🟥 J4s	🟥 J3s	🟥 J2s
🟩 ATo	🟩 KTo	🟩 QTo	🟩 JTo	🟩 TT	🟩 T9s	🟨 T8s	🟨 T7s	🟥 T6s	🟥 T5s	🟥 T4s	🟥 T3s	🟥 T2s
🟩 A9o	🟨 K9o	🟨 Q9o	🟨 J9o	🟨 T9o	🟩 99	🟨 98s	🟨 97s	🟥 96s	🟥 95s	🟥 94s	🟥 93s	🟥 92s
🟨 A8o	🟨 K8o	🟥 Q8o	🟥 J8o	🟥 T8o	🟥 98o	🟩 88	🟨 87s	🟥 86s	🟥 85s	🟥 84s	🟥 83s	🟥 82s
🟨 A7o	🟥 K7o	🟥 Q7o	🟥 J7o	🟥 T7o	🟥 97o	🟥 87o	🟩 77	🟥 76s	🟥 75s	🟥 74s	🟥 73s	🟥 72s
🟨 A6o	🟥 K6o	🟥 Q6o	🟥 J6o	🟥 T6o	🟥 96o	🟥 86o	🟥 76o	🟩 66	🟥 65s	🟥 64s	🟥 63s	🟥 62s
🟨 A5o	🟥 K5o	🟥 Q5o	🟥 J5o	🟥 T5o	🟥 95o	🟥 85o	🟥 75o	🟥 65o	🟨 55	🟥 54s	🟥 53s	🟥 52s
🟨 A4o	🟥 K4o	🟥 Q4o	🟥 J4o	🟥 T4o	🟥 94o	🟥 84o	🟥 74o	🟥 64o	🟥 54o	🟨 44	🟥 43s	🟥 42s
🟨 A3o	🟥 K3o	🟥 Q3o	🟥 J3o	🟥 T3o	🟥 93o	🟥 83o	🟥 73o	🟥 63o	🟥 53o	🟥 43o	🟥 33	🟥 32s
🟥 A2o	🟥 K2o	🟥 Q2o	🟥 J2o	🟥 T2o	🟥 92o	🟥 82o	🟥 72o	🟥 62o	🟥 52o	🟥 42o	🟥 32o	🟥 22

Figure 4.2: The button strategy played by (tight) *Poki* displayed in a preflop hand matrix. Raising hands are highlighted in green, calling hands in yellow, and folding hands in red.

🟩 AA	🟩 AKs	🟩 AQs	🟩 AJs	🟩 ATs	🟩 A9s	🟩 A8s	🟩 A7s	🟩 A6s	🟩 A5s	🟩 A4s	🟩 A3s	🟨 A2s
🟩 AKo	🟩 KK	🟩 KQs	🟩 KJs	🟩 KTs	🟩 K9s	🟩 K8s	🟨 K7s	🟨 K6s	🟨 K5s	🟨 K4s	🟨 K3s	🟨 K2s
🟩 AQo	🟩 KQo	🟩 QQ	🟩 QJs	🟩 QTs	🟩 Q9s	🟨 Q8s	🟨 Q7s	🟨 Q6s	🟨 Q5s	🟨 Q4s	🟨 Q3s	🟨 Q2s
🟩 AJo	🟩 KJo	🟩 QJo	🟩 JJ	🟩 JTs	🟩 J9s	🟨 J8s	🟨 J7s	🟨 J6s	🟨 J5s	🟨 J4s	🟨 J3s	🟨 J2s
🟩 ATo	🟩 KTo	🟩 QTo	🟩 JTo	🟩 TT	🟩 T9s	🟨 T8s	🟨 T7s	🟨 T6s	🟨 T5s	🟨 T4s	🟥 T3s	🟥 T2s
🟩 A9o	🟨 K9o	🟨 Q9o	🟨 J9o	🟨 T9o	🟩 99	🟨 98s	🟨 97s	🟨 96s	🟨 95s	🟥 94s	🟥 93s	🟥 92s
🟨 A8o	🟨 K8o	🟨 Q8o	🟨 J8o	🟨 T8o	🟨 98o	🟩 88	🟨 87s	🟨 86s	🟨 85s	🟥 84s	🟥 83s	🟥 82s
🟨 A7o	🟨 K7o	🟨 Q7o	🟨 J7o	🟨 T7o	🟨 97o	🟨 87o	🟩 77	🟨 76s	🟨 75s	🟥 74s	🟥 73s	🟥 72s
🟨 A6o	🟨 K6o	🟨 Q6o	🟥 J6o	🟥 T6o	🟥 96o	🟥 86o	🟥 76o	🟩 66	🟨 65s	🟥 64s	🟥 63s	🟥 62s
🟨 A5o	🟨 K5o	🟥 Q5o	🟥 J5o	🟥 T5o	🟥 95o	🟥 85o	🟥 75o	🟥 65o	🟨 55	🟨 54s	🟥 53s	🟥 52s
🟨 A4o	🟨 K4o	🟥 Q4o	🟥 J4o	🟥 T4o	🟥 94o	🟥 84o	🟥 74o	🟥 64o	🟥 54o	🟨 44	🟥 43s	🟥 42s
🟨 A3o	🟨 K3o	🟥 Q3o	🟥 J3o	🟥 T3o	🟥 93o	🟥 83o	🟥 73o	🟥 63o	🟥 53o	🟥 43o	🟨 33	🟥 32s
🟨 A2o	🟥 K2o	🟥 Q2o	🟥 J2o	🟥 T2o	🟥 92o	🟥 82o	🟥 72o	🟥 62o	🟥 52o	🟥 42o	🟥 32o	🟥 22

Figure 4.3: The button strategy played by (loose) *Poki* displayed in a preflop hand matrix. Raising hands are highlighted in green, calling hands in yellow, and folding hands in red.

🟩 AA	🟩 AKs	🟩 AQs	🟩 AJs	🟩 ATs	🟩 A9s	🟩 A8s	🟩 A7s	🟩 A6s	🟩 A5s	🟩 A4s	🟩 A3s	🟩 A2s
🟩 AKo	🟩 KK	🟩 KQs	🟩 KJs	🟩 KTs	🟩 K9s	🟩 K8s	🟩 K7s	🟩 K6s	🟩 K5s	🟩 K4s	🟩 K3s	🟩 K2s
🟩 AQo	🟩 KQo	🟩 QQ	🟩 QJs	🟩 QTs	🟩 Q9s	🟩 Q8s	🟩 Q7s	🟩 Q6s	🟩 Q5s	🟥 Q4s	🟥 Q3s	🟥 Q2s
🟩 AJo	🟩 KJo	🟩 QJo	🟩 JJ	🟩 JTs	🟩 J9s	🟩 J8s	🟩 J7s	🟥 J6s	🟥 J5s	🟥 J4s	🟥 J3s	🟥 J2s
🟩 ATo	🟩 KTo	🟩 QTo	🟩 JTo	🟩 TT	🟩 T9s	🟩 T8s	🟩 T7s	🟥 T6s	🟥 T5s	🟥 T4s	🟥 T3s	🟥 T2s
🟩 A9o	🟩 K9o	🟩 Q9o	🟩 J9o	🟩 T9o	🟩 99	🟩 98s	🟩 97s	🟥 96s	🟥 95s	🟥 94s	🟥 93s	🟥 92s
🟩 A8o	🟥 K8o	🟥 Q8o	🟥 J8o	🟩 T8o	🟩 98o	🟩 88	🟩 87s	🟩 86s	🟥 85s	🟥 84s	🟥 83s	🟥 82s
🟩 A7o	🟥 K7o	🟥 Q7o	🟥 J7o	🟥 T7o	🟥 97o	🟥 87o	🟩 77	🟩 76s	🟩 75s	🟥 74s	🟥 73s	🟥 72s
🟩 A6o	🟥 K6o	🟥 Q6o	🟥 J6o	🟥 T6o	🟥 96o	🟥 86o	🟥 76o	🟩 66	🟩 65s	🟥 64s	🟥 63s	🟥 62s
🟩 A5o	🟥 K5o	🟥 Q5o	🟥 J5o	🟥 T5o	🟥 95o	🟥 85o	🟥 75o	🟥 65o	🟩 55	🟥 54s	🟥 53s	🟥 52s
🟩 A4o	🟥 K4o	🟥 Q4o	🟥 J4o	🟥 T4o	🟥 94o	🟥 84o	🟥 74o	🟥 64o	🟥 54o	🟩 44	🟥 43s	🟥 42s
🟩 A3o	🟥 K3o	🟥 Q3o	🟥 J3o	🟥 T3o	🟥 93o	🟥 83o	🟥 73o	🟥 63o	🟥 53o	🟥 43o	🟩 33	🟥 32s
🟥 A2o	🟥 K2o	🟥 Q2o	🟥 J2o	🟥 T2o	🟥 92o	🟥 82o	🟥 72o	🟥 62o	🟥 52o	🟥 42o	🟥 32o	🟩 22

Figure 4.4: A button strategy displayed in a preflop hand matrix suggested by *stoxtrader*'s book. Raising hands are highlighted in green while folding hands are highlighted in red.

4.4 Solution Technique

The technique used to generate strategies for HUEs involves using CFR with special features: forcing actions and seeding players with strategies.

4.4.1 Forced-Action and Strategy-Seeded CFR

Typically, with 3-player CFR, there are no assumptions about the strategies of the three players. The players begin by playing completely randomly and iteratively adjust the frequencies with which they take certain actions based on how much regret was associated with those actions in the past. Each hand, the players are dealt their private hole cards, play according to their current distribution over the possible actions, and then update their regret.

If we like, however, we could seed one or more of the players with an **initial** or **static strategy**. With an *initial strategy*, the player begins with the given strategy (instead of random) and iteratively modifies the strategy based on the regret. With a *static strategy*, the strategy is never updated based on regret.

Another feature which can be added to CFR is **forced actions**[1], an action sequence we would like to force beginning at the root node. With *forced actions*, only iterations in which the pre-specified action sequence occurs are used to update regret, otherwise the iteration is discarded. On each new iteration, private hole cards are dealt to each player who takes an action based on the hole cards and the corresponding bucket's action probabilities (beginning with the button). For example, if the *forced action* was raise-fold (rf), after being dealt cards, the button takes an action based on its current strategy. If the action taken is fold or call instead of raise then the iteration is discarded and a new hand is dealt. If the action taken is raise, however, then the small blind takes an action based on its current strategy. If the action taken by the small blind is call or raise instead of fold, then the iteration is discarded. If the action taken is fold, then the remainder of the hand is played out

[1]While we call these "forced actions", they can be interpreted as "preferred actions" since we *force* CFR to discard any iteration where *preferred* actions are not taken.

between the button and the big blind based on their strategies and regrets are updated at the end of the hand.

4.4.2 Heads-Up Subgame Solution

Let us call a CFR computation using both of the *forced actions* and *strategy seeding* (initial or static) features **forced-action strategy-seeded counterfactual regret minimization** (FASSCFR). Notice that if the forced action sequences used are suffixed by a fold action, then the hand becomes heads-up between the two active players. This situation always includes the same initial pot size and can be solved as a two-player zero-sum game. This is quite convenient as we have theoretical guarantees (for perfect recall) that we will find an ϵ-Nash equilibrium.

Although we are guaranteed to find an equilibrium for the HUE subgame, the resulting subgame strategy may not be part of the 3-player equilibrium since we must seed the strategy. However, this does not suggest that the HUE strategy cannot provide an improvement over the base player strategy. Remember that the 3-player game is over one *million* times as large as a heads-up game! This means that much larger abstractions can be used for the HUEs than were used to generate the base player's 3-player strategy.

There are infinitely many strategies with which to seed a HUE. Two natural ways to seed the HUEs are

- **Uniform strategy-seeding**. This naive approach to seeding assumes that the players take all of the actions in the *forced action* sequence 100% of the time. That is, the distribution over hands or buckets of all active players is completely uniform. For example, if the *forced action* was raise-call-fold (rcf) then we would seed the button with a strategy of raising 100% of the time, the small blind with calling 100% of the time, and the big blind with folding 100% of the time.

- **Expert knowledge strategy-seeding**. Instead of uniformly seeding the strategies, we could devise a sensible preflop strategy based on expert knowledge (advice from Poker professionals and/or based on strong computer agents such as those generated in Chapter 3). For example, we could seed the button with a strategy of raising 40% of the time (top two buckets in a 5-bucket abstraction) and folding 60% of the time (bottom 3 buckets) to correspond with the expert knowledge from professional Poker players provided in Table 4.4. Thus, when the action is raise-fold (rf) then the big blind can deduce that the button is only playing the top 40% of hands. It should be noted, however, that if an opponent plays a drastically different distribution than that assumed, poor play could result.

The next subsection contains the strategies used to seed the HUEs in this dissertation. In Section 4.6 of this chapter, we will present experimental tournament results for HUEs generated with 5-bucket perfect recall abstractions using both *uniform* and *expert knowledge strategy-seeding*. The base players using *uniform strategy-seeding* will have their names prefixed by "Uni_" while the base players using *expert knowledge strategy-seeding* will have their names prefixed by "Exp_". The *expert knowledge strategy-seeding* did not include preflop strategies used by the agents from Chapter 3 because the preflop strategy distributions for the 3-player CFR-generated strategies were not available when the HUE strategies were generated.

4.4.3 Heads-Up Expert Strategies

The heads-up experts (HUEs) generated for this dissertation were not intended to be perfect, by any means. The intention of incorporating and testing the HUEs was to determine whether fairly abstract and naively created strategies could be used to improve play; they were simply a proof of concept. Thus, for convenience, we constructed HUEs with the following properties:

- When using a one-dimensional metric to bucket hands, it is conventional to place hands with higher metric values in higher numbered buckets. We adopt this convention here; namely, we have a bucket ranking $5 > 4 > 3 > 2 > 1$ for our 5-bucket $E[HS^2]$ abstraction.
- We have used pure prior distributions: the probability of taking an action in a bucket is either 0 or 1.
- The highest (i.e. strongest) buckets are used for raise actions, the next highest buckets are used for call actions, and the lowest buckets are folded.

Something to consider for future work is that using mixed strategy priors and not having a strict ordering of buckets corresponding to actions (e.g. raise buckets > call buckets > fold buckets) may lead to better play. These changes would be able to better take into account *bluffing* (playing weak hands strongly) and *slow-playing* (playing strong hands weakly).

Figures 4.2 to 4.4 show examples of strategies in the full game; however, in practice, we must use strategies from an abstract game. Tables 4.5 to 4.7 show some abstract game strategies that can be used to seed HUEs. The titles of the last three columns represent the positions (button, small blind, and big blind) and the entries correspond to the buckets for which the player takes the corresponding row-action. For example, the way to interpret the "rrf" column in Table 4.6 is that the button will raise ('r') with buckets 3-5, the small blind will reraise ('r') with bucket 5, and the big blind will fold ('f') buckets 1-4.

Table 4.5 was used for the uniform strategy-seeded HUEs and Table 4.6 was used for the expert knowledge strategy-seeded HUEs. Table 4.7 represents some possible strategies to seed future 5-bucket HUEs. The priors for actions shared between different betting sequences is more consistent with these strategies. For instance, notice that in Table 4.6 that the button raises buckets 4-5 in the "rf" betting sequence yet raises buckets 3-5 in "rrf" and "rcf".

Bet Sequence	BTN	SB	BB
f	1-5	–	–
rf	1-5	1-5	–
rrf	1-5	1-5	1-5
rcf	1-5	1-5	1-5
cf	1-5	1-5	–
crf	1-5	1-5	1-5

Table 4.5: The uniform strategy used in this dissertation to seed the six 5-bucket heads-up experts.

Bet Sequence	BTN	SB	BB
f	1-2	–	–
rf	4-5	1-4	–
rrf	3-5	5	1-4
rcf	3-5	4	1-2
cf	3	1-2	–
crf	3	5	1-2

Table 4.6: The expert knowledge strategy used in this dissertation to seed the six 5-bucket heads-up experts.

4.5 Master Agent

Once a set of HUEs has been generated using FASSCFR, we require a way to piece their corresponding subgames back together so that the heads-up strategies can be played in the 3-player game. We want to play a 3-player base strategy until we encounter a heads-up scenario that we have pre-solved using FASSCFR. To do this, we'll use a *master agent* whose sole purpose is to summon different agents depending on the bet sequence of the current hand. The **master agent** can be likened to a *coach* who decides when to field or bench a player (except that the decisions are much more automatic in this case).

4.5.1 Implementation

There are two basic ways in which the *master agent* can be implemented, each with its own advantages and disadvantages:

- The *master agent* loads the base player and all HUEs into memory. The base player is always played when there are three active players. If, however, one of the HUE's priming bet sequence occurs (and thus one of the players has folded), that HUE is summoned and is played for the remainder of the hand.
- The *master agent* simply becomes a conceptual idea as the HUE strategies actually replace the corresponding subgame strategies of the base player in a new strategy file.

The advantage of using the first method is being able to use a base player for which we have no explicit strategy file (i.e. the base player acts as a sort of "black box" where the input is the game state and the output is its corresponding action). The disadvantages of using this method

Bet Sequence	BTN	SB	BB
f	1-2	–	–
rf	4-5	1-3	–
rrf	4-5	5	1-3
rcf	4-5	4	1-2
cf	3	1-2	–
crf	3	5	1-2

Table 4.7: A potential expert knowledge strategy to seed the six 5-bucket heads-up experts as seen in Figure 4.1.

include the requirement for sufficient memory to preload all of the agents and that actual gameplay involving the *master agent* will be slower as it needs to query the strategy of different players quite often. The second method does not suffer from slower play or memory issues (as long as there is enough memory to load the base player) but does not allow for "black box" agents as base players. The first method was implemented for the *master agents* discussed in this thesis simply due to the convenience of its implementation and its ability to allow us to use *Poki* as the base player.

4.6 Experimental Results

A tournament similar to Section 3.5.3 was set up to evaluate the HUE agents. More specifically, there were exactly 7 entrants and each of the $\binom{7}{3} = 35$ three-player matchups consisted of running 6 positional permutations of 200,000 hands (1.2 million hands total). However, this tournament did not include any chumps (*Always-Call*, *Always-Raise*, and *Probe*) as entrants. There were two main reasons for this decision: removing chumps should create a playing field more comparable to those in the Annual Computer Poker Competition and including just three more entrants results in $\binom{10}{3} = 120$ three-player matchups as opposed to 35 matchups with 7 entrants. The set of agents entered into the tournament include

- *Poki*
- *Multi2*
- *Uni_Multi2* (Uni_M2)
- *Exp_Multi2* (Exp_M2)
- *IR16*
- *Uni_IR16*
- *Exp_IR16*

Only IR16_43M was entered into this tournament since the results of *IR16_20M* and *IR16_43M* were quite similar in Section 3.5.3 and including both of the *IR16* agents in the tournament would create many more matchups. All three of the *IR16* agents listed above implicitly use *IR16_43M* as the base player.

	Poki, Multi2	Poki, Uni_M2	Poki, Exp_M2	Poki, IR16	Poki, Uni_IR16	Poki, Exp_IR16	Multi2, Uni_M2	Multi2, Exp_M2	Multi2, IR16	Multi2, Uni_IR16	Multi2, Exp_IR16	Uni_M2, Exp_M2	Uni_M2, IR16	Uni_M2, Uni_IR16	Uni_M2, Exp_IR16	Exp_M2, IR16	Exp_M2, Uni_IR16	Exp_M2, Exp_IR16	IR16, Uni_IR16	IR16, Exp_IR16	Uni_IR16, Exp_IR16	Winrate
Poki							-0.034	-0.050	-0.065	-0.050	-0.059	-0.064	-0.075	-0.060	-0.072	-0.092	-0.080	-0.092	-0.089	-0.106	-0.090	-0.072
Multi2		-0.001	-0.010	-0.030	-0.020	-0.027						-0.031	-0.057	-0.042	-0.053	-0.067	-0.053	-0.062	-0.083	-0.090	-0.081	-0.047
Uni_M2	0.035		0.014	-0.009	0.007	-0.006		0.004	-0.025	-0.014	-0.019					-0.043	-0.033	-0.035	-0.061	-0.069	-0.055	-0.021
Exp_M2	0.060	0.050		0.029	0.037	0.030	0.028		0.002	0.005	0.004		-0.013	-0.006	-0.010				-0.032	-0.036	-0.025	0.008
Uni_IR16	0.070	0.054	0.043	0.019		0.022	0.057	0.049	0.023		0.028	0.039	0.009		0.013	-0.002		0.000		-0.029		0.026
Exp_IR16	0.086	0.078	0.063	0.050	0.067		0.071	0.058	0.039	0.053		0.045	0.030	0.042		0.015	0.025		0.009			0.049
IR16	0.095	0.084	0.063		0.070	0.056	0.081	0.065		0.060	0.051	0.055		0.053	0.040		0.034	0.022			0.020	0.057

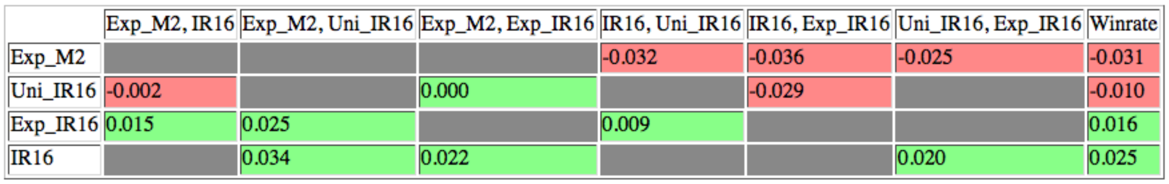
Figure 4.5: A cross-table showing the results of a 3-player HUE tournament with 7 agents. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (last column) are within ± 0.002 with 95% confidence.

	Multi2, Uni_M2	Multi2, Exp_M2	Multi2, IR16	Multi2, Uni_IR16	Multi2, Exp_IR16	Uni_M2, Exp_M2	Uni_M2, IR16	Uni_M2, Uni_IR16	Uni_M2, Exp_IR16	Exp_M2, IR16	Exp_M2, Uni_IR16	Exp_M2, Exp_IR16	IR16, Uni_IR16	IR16, Exp_IR16	Uni_IR16, Exp_IR16	Winrate
Multi2						-0.031	-0.057	-0.042	-0.053	-0.067	-0.053	-0.062	-0.083	-0.090	-0.081	-0.062
Uni_M2		0.004	-0.025	-0.014	-0.019					-0.043	-0.033	-0.035	-0.061	-0.069	-0.055	-0.035
Exp_M2	0.028		0.002	0.005	0.004		-0.013	-0.006	-0.010				-0.032	-0.036	-0.025	-0.008
Uni_IR16	0.057	0.049	0.023		0.028	0.039	0.009		0.013	-0.002		0.000		-0.029		0.019
Exp_IR16	0.071	0.058	0.039	0.053		0.045	0.030	0.042		0.015	0.025		0.009			0.039
IR16	0.081	0.065		0.060	0.051	0.055		0.053	0.040		0.034	0.022			0.020	0.048

Figure 4.6: A cross-table showing the results of the remaining 6 agents in a 3-player HUE *Equilibrium* competition with 7 agents after eliminating the worst player. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (last column) are within ± 0.003 with 95% confidence.

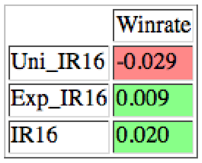
	Uni_M2, Exp_M2	Uni_M2, IR16	Uni_M2, Uni_IR16	Uni_M2, Exp_IR16	Exp_M2, IR16	Exp_M2, Uni_IR16	Exp_M2, Exp_IR16	IR16, Uni_IR16	IR16, Exp_IR16	Uni_IR16, Exp_IR16	Winrate
Uni_M2					-0.043	-0.033	-0.035	-0.061	-0.069	-0.055	-0.049
Exp_M2		-0.013	-0.006	-0.010				-0.032	-0.036	-0.025	-0.020
Uni_IR16	0.039	0.009		0.013	-0.002		0.000		-0.029		0.005
Exp_IR16	0.045	0.030	0.042		0.015	0.025		0.009			0.028
IR16	0.055		0.053	0.040		0.034	0.022			0.020	0.037

Figure 4.7: A cross-table showing the results of the remaining 5 agents in a 3-player HUE *Equilibrium* competition with 7 agents after eliminating the two worst players. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (last column) are within ± 0.004 with 95% confidence.



	Exp_M2, IR16	Exp_M2, Uni_IR16	Exp_M2, Exp_IR16	IR16, Uni_IR16	IR16, Exp_IR16	Uni_IR16, Exp_IR16	Winrate
Exp_M2				-0.032	-0.036	-0.025	-0.031
Uni_IR16	-0.002		0.000		-0.029		-0.010
Exp_IR16	0.015	0.025		0.009			0.016
IR16		0.034	0.022			0.020	0.025

Figure 4.8: A cross-table showing the results of the remaining 4 agents in a 3-player HUE *Equilibrium* competition with 7 agents after eliminating the three worst players. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The total winrates (last column) are within ± 0.005 with 95% confidence.



	Winrate
Uni_IR16	-0.029
Exp_IR16	0.009
IR16	0.020

Figure 4.9: A cross-table showing the final three agents in a 3-player HUE *Equilibrium* competition with 7 agents after eliminating the worst player. The descending rows are ordered by increasing *total bankroll* winrates. Units are provided in sb/h. The winrates are within ± 0.008 with 95% confidence.

Tables 4.5 to 4.9 show the results for the tournament. All individual matchup winrates are within ± 0.009 sb/h with 95% confidence. The confidence intervals of the total winrates are provided in the captions. For comparison, the *Always-Fold* strategy loses 0.500 sb/h in a 3-player game. The rankings for the *Bankroll* competition are

1. *IR16*
2. *Exp_IR16*
3. *Uni_IR16*
4. *Exp_Multi2*
5. *Uni_Multi2*
6. *Multi2*
7. *Poki*

There are several things that we can notice from these results. The rankings for both the *Equilibrium* and *Bankroll* competitions are identical. In the tournament results from Chapter 3, we saw that *Multi2* did very well in the *Bankroll* competition as it exploited chumps more than any other agent. In this tournament, however, only strong agents were included. We can see that the top two bankroll players (*IR16* and *Exp_IR16*) win in every single matchup they participate in while the opposite is true for the worst two bankroll players (*Poki* and *Multi2*).

Although Poker is a nontransitive game, it is interesting to examine the relative trends in results from Table 4.5. Namely, as you move down the rows of a given column (corresponding to higher-ranked bankroll players), the winrates increase monotonically with a few exceptions. In addition to this trend, as we move across the columns of a given row (generally corresponding to higher-ranked bankroll players although exactly), the winrates decrease. Notice that having all AI agents (instead of including chumps) causes the results to be much more predictable. Another trend – related to those just described – is that every agent loses the most against the top two bankroll players (excluding themself) and every agent wins the most against the worst two bankroll players (excluding themself). The regularity of these trends seems to be the reason we obtain the same rankings from both the *Equilibrium* and *Bankroll* competitions.

Note also that the *expert knowledge strategy-seeded* HUEs did better than the *uniform strategy-seeded* HUEs when either *Multi2* or *IR16* were used as the base player. This result is not too surprising as the assumption that the opponent can hold a completely random hand after several betting actions have taken place is a rather poor one.

It is interesting that *Multi2*'s performance has been improved by adding HUEs while *IR16* performs worse when HUEs are added. This result demonstrates just how effective using CFR can be for generating very abstract, imperfect recall, 3-player agents despite having no theoretical guarantees. Although adding the expert strategy-seeded HUEs did not improve play over *IR16*, we can still see that expert seeding was significantly better than the uniform strategy-seeding HUEs.

Notice the similarities between the HUEs used here and the *PsOpti* agents described in Section 2.3 and in [2]. The uniform strategy-seeding is similar to the way *PsOpti1* knits strategies together: a subgame strategy generated assuming a uniform distribution over hands is appended to a preflop strategy (in this case whichever base player is playing). The expert knowledge strategy-seeding is similar to the way *PsOpti2* knits strategies together: a preflop strategy is used to provide conditional probabilities as input to generate the subgame strategy.

Since the HUEs and base players were being generated simultaneously with CFR, we were unable to use the base players' strategies to seed the HUEs. We believe that strategy-seeding based on *IR16*'s preflop play would out-perform *IR16*, and this is left as future work (Section 6.1).

Chapter 5

Computer Poker Competition Results

The Annual Computer Poker Competition started in 2006 and has since been held once per year. The objective of the competition is that Artificial Intelligence research will benefit from using Poker – a stochastic imperfect information game resembling real-world situations – as its testbed.

The competition currently consists of three Texas Hold'em variants: heads-up limit, heads-up no-limit, and ring limit. For each game type, there are two competitions (or sub-competitions) evaluated with duplicate matches: *Bankroll* and *Equilibrium*. The goal of the *Bankroll* competition is to win as many chips as possible rewarding exploitative play. The goal of the *Equilibrium* competition is to win as many chips as possible against stronger and stronger sets of opponents. That is, the worst-ranked player from the *Bankroll* competition is eliminated and the players are re-ranked based on their winrates against all players but the worst. This process continues until all players have been eliminated.

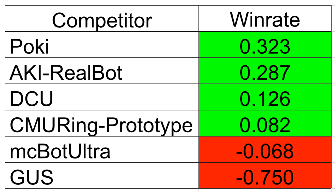
In this chapter, we will present the limit ring results from the 2008 and 2009 Computer Poker Competitions.

5.1 2008 AAAI Computer Poker Ring Competition

The results of the 2008 Computer Poker Competition were announced at the AAAI conference on July 15, 2008 in Chicago, Illinois. In total there were 12 competitors from 6 different countries and 19 agents submitted (distributed amongst the three sub-competitions: heads-up limit, heads-up no-limit, ring limit).

The ring limit competition was a 6-player limit Hold'em format. There were 6 agents submitted from 5 different teams (Technical University Darmstadt submitted two). Due to just having sufficient numbers of players for one table, there was only a *Bankroll* competition and no way to run an *Equilibrium* competition (as the agents were not required to be able to play in a format with less than 5 players).

Results are presented in Table 5.1 in small bets per hand (sb/h) with the top-ranked player at the top. A total of 440,000 hands were played with random permutations of the relative positions. Unfortunately, standard deviations for the winrates were not provided with the results of this competition. For comparison, the *Always-Fold* strategy loses 0.250 sb/h in a 6-player game.



Competitor	Winrate
Poki	0.323
AKI-RealBot	0.287
DCU	0.126
CMURing-Prototype	0.082
mcBotUltra	-0.068
GUS	-0.750

Figure 5.1: The ranking and results of competitors in the limit ring 2008 Computer Poker Competition. Units are presented in small bets per hand (sb/h).

5.2 2009 IJCAI Computer Poker Ring Competition

The results of the 2009 Computer Poker Competition were announced at the IJCAI conference on July 15, 2009 in Pasadena, California. In total, there were 14 competitors (10 universities and 4 independents) from 7 different countries and 25 agents submitted (13 for heads-up limit, 5 for heads-up no-limit, and 7 for ring limit).

The ring limit competition this year was reduced from a 6-player to 3-player limit Hold'em format since the 3-player game is a challenging enough problem. Each 3-player matchup involved 660,000 hands (6 position permutations by 110 matches of 1000 hands). There were 7 agents submitted from 5 different teams. Matchups including two agents from the same team were not used in determining the winners of the competitions.

In alphabetical order, the competitors of the 3-player 2009 Computer Poker Competition included

- *akuma*
- *Bluechip* (Blue)
- *CMURingLimit* (CMU)
- *dcu3pl* (dcu)
- *dpp*
- *Hyperborean-BR* (H-BR)
- *Hyperborean-Eqm* (H-Eqm)

The *akuma* and *dpp* competitors were part of the same team, Technical University Darmstadt. *Hyperborean-BR* and *Hyperborean-Eqm* were submitted by the author as part of the University of Alberta CPRG team. *Hyperborean-BR* was the *Exp_IR16* agent described in Chapter 4 and *Hyperborean-Eqm* was the *IR16_43M* agent described in Chapter 3.

The 7-competitor cross-table has been divided into separate tables, Tables 5.2 to 5.6, for each of the five teams and are presented in the order of their ranking in the *Bankroll* competition. Results are provided in small bets per hand (sb/h). For comparison, the *Always-Fold* strategy loses 0.500 sb/h in a 3-player game. Standard deviation values were given for each matchup but were not included in the tables (due to space). However, the minimum and maximum 95% confidence interval sizes have been provided in each table's caption.

	dpp, dcu	dpp, Blue	dpp, CMU	akuma, dcu	akuma, Blue	akuma, CMU	dcu, Blue	dcu, CMU	Blue, CMU	Winrate
H-Eqm	0.257	0.384	0.271	0.217	0.374	0.231	0.442	0.255	0.438	0.319
H-BR	0.224	0.360	0.235	0.19	0.364	0.215	0.426	0.235	0.440	0.299

Figure 5.2: The 3-player 2009 Computer Poker Competition results for the University of Alberta agents, *Hyperborean-BR* and *Hyperborean-Eqm*, against all pairs of opponents (from different teams). Units are presented in small bets per hand (sb/h). All results are within ± 0.006 to 0.008 with 95% confidence.

	dcu, H-Eqm	dcu, H-BR	dcu3pl, Bluechip	dcu, CMU	H-Eqm, Blue	H-Eqm, CMU	H-BR, Blue	H-BR, CMU	Blue, CMU	Winrate
dpp	0.010	0.037	0.590	0.149	0.284	-0.128	0.304	-0.105	0.401	0.171
akuma	0.005	0.024	0.359	0.198	0.195	0.019	0.200	0.028	0.332	0.151

Figure 5.3: The 3-player 2009 Computer Poker Competition results for the Technical University Darmstadt agents, *akuma* and *dpp*, against all pairs of opponents (from different teams). Units are presented in small bets per hand (sb/h). All results are within ± 0.006 to 0.010 with 95% confidence.

	dpp, dcu	dpp, H-Eqm	dpp, H-BR	dpp, Blue	akuma, dcu	akuma, H-Eqm	akuma, H-BR	akuma, Blue	dcu, H-Eqm	dcu, H-BR	dcu, Blue	H-Eqm, Blue	H-BR, Blue	Winrate
CMU	0.015	-0.141	-0.128	0.216	-0.096	-0.249	-0.242	0.095	-0.133	-0.128	0.199	0.048	0.054	-0.037

Figure 5.4: The 3-player 2009 Computer Poker Competition results for the *CMURingLimit* agent. Units are presented in small bets per hand (sb/h). All results are within ± 0.006 to 0.010 with 95% confidence.

The rankings for the *Bankroll* competition are

1. *Hyperborean-Eqm*
2. *Hyperborean-BR*
3. *dpp*
4. *akuma*
5. *CMURingLimit*
6. *dcu3pl*
7. *Bluechip*

The rankings for the *Equilibrium* competition are

1. *Hyperborean-Eqm*

	dpp, H-Eqm	dpp, H-BR	dpp, Blue	dpp, CMU	akuma, H-Eqm	akuma, H-BR	akuma, Blue	akuma, CMU	H-Eqm, Blue	H-Eqm, CMU	H-BR, Blue	H-BR, CMU	Blue, CMU	Winrate
dcu	-0.267	-0.260	0.060	-0.164	-0.221	-0.213	0.129	-0.100	0.086	-0.121	0.102	-0.106	0.255	-0.063

Figure 5.5: The 3-player 2009 Computer Poker Competition results for the *dcu3pl* agent. Units are presented in small bets per hand (sb/h). All results are within ± 0.006 to 0.008 with 95% confidence.

	dpp, dcu	dpp, H-Eqm	dpp, H-BR	dpp, CMU	dpp, dcu	akuma, H-Eqm	akuma, H-BR	akuma, CMU	dcu, H-Eqm	dcu, H-BR	dcu, CMU	H-Eqm, CMU	H-BR, CMU	Winrate
Blue	-0.649	-0.667	-0.662	-0.616	-0.487	-0.568	-0.563	-0.426	-0.527	-0.526	-0.452	-0.486	-0.493	-0.548

Figure 5.6: The 3-player 2009 Computer Poker Competition results for the *Bluechip* agent. Units are presented in small bets per hand (sb/h). All results are within ± 0.008 to 0.012 with 95% confidence.

Competitor	Winrate
HyperboreanRing-Eqm	0.319
HyperboreanRing-BR	0.299
dpp	0.171
akuma	0.151
CMURingLimit	-0.037
dcu3pl	-0.063
Bluechip	-0.548

Figure 5.7: The 3-player 2009 Computer Poker Competition results for the *Bankroll* competition and the *Equilibrium* competition with 7 players left. Units are presented in small bets per hand (sb/h). All results are within ± 0.004 with 95% confidence.

Competitor	Winrate
HyperboreanRing-Eqm	0.246
HyperboreanRing-BR	0.220
akuma	0.055
dpp	-0.006
CMURingLimit	-0.137
dcu3pl	-0.181

Figure 5.8: The 3-player 2009 Computer Poker Competition results for the *Equilibrium* competition with 6 players left. Units are presented in small bets per hand (sb/h). All results are within ± 0.006 with 95% confidence.

Competitor	Winrate
HyperboreanRing-Eqm	0.251
HyperboreanRing-BR	0.225
akuma	0.024
dpp	-0.116
CMURingLimit	-0.190

Figure 5.9: The 3-player 2009 Computer Poker Competition results for the *Equilibrium* competition with 5 players left. Units are presented in small bets per hand (sb/h). All results are within ± 0.008 with 95% confidence.

2. *Hyperborean-BR*
3. *akuma*
4. *dpp*
5. *CMURingLimit*
6. *dcu3pl*
7. *Bluechip*

Notice that the only difference between the rankings for the *Bankroll* and *Equilibrium* competitions is that *akuma* and *dpp* switch places in third and fourth. The reason for terminating the *equilibrium* competition run-off at 5 players was due to the University of Alberta and Technical University Darmstadt each having two remaining agents. Since agents from the same team were not included in the tournament, the top 5 rankings for the *Equilibrium* competition were determined from the rankings with 5 players left.

Chapter 6

Conclusions

In this thesis, we've shown that

- **CFR was used to generate a very abstract perfect recall 3-player agent, *Multi2*, that out-performs *Poki*, the 2008 ring limit Computer Poker Competition champion.** Although there are no theoretical guarantees for CFR regarding multiplayer games, we showed that CFR can still be used to generate very strong agents (even for a very abstract game). *Multi2* uses a 2-bucket E$[HS^2]$ abstraction. This work represents the first research performed with CFR on multiplayer games.

- **CFR was used to generate two-player ϵ-Nash equilbria, called heads-up experts (HUEs), corresponding to 3-player subgames wherein one of the players folds preflop.** Over one half of 3-player hands among competent players result in one player folding preflop. Since the resulting subgame is a two-player zero-sum game, we can create a heads-up expert by computing an ϵ-Nash equilibrium with CFR. We constructed HUEs using a 5-bucket E$[HS^2]$ abstraction using different strategies to seed play until a fold occurs. We seeded one strategy with a uniform distribution (i.e. we assume the players hold a random hand), prefixed with "Uni_" and seeded a second strategy with expert knowledge, prefixed by "Exp_".

- **Master agents, *Uni_Multi2* and *Exp_Multi2*, using *Multi2* as a base player out-perform *Multi2*.** A master agent acts as a sort of "coach" that fields players where appropriate. In our case, the master agent always fields a base player when there are three active players and substitutes a HUE if its defining bet sequence occurs preflop (one of *f, rf, rrf, rcf, cf, crf*). Using both uniform and expert knowledge strategy seeding with *Multi2* as the base player improved the winrate of the master agent over the base player alone.

- **CFR was used to generate a very abstract imperfect recall 3-player agent, *IR16*, that out-performs *Poki* and all of the agents that use *Multi2* as a base player.** An agent, *IR16*, with approximately the same size of game state space as *Multi2* was constructed. This agent forgets its previous round buckets and has a 16-bucket E$[HS^2]$ abstraction. This is the strongest agent described in this dissertation.

- **Two of the agents created, *IR16* and *Exp_IR16*, are currently the best 3-player computer Poker agents in the world.** The Annual Computer Poker Competition (CPC) has two ring limit competitions: *Bankroll* and *Equilibrium*. The 2009 CPC's ring games had a 3-player format (compared to 6-player in 2008). The submitted agents placed first and second, respectively, in both competitions: *IR16* placed first in both and *Exp_IR16* placed second in both.

6.1 Future Work

Since the experiments required for this work were very time-consuming (due to the size of the 3-player game) and were often dependent on the completion of other experiments, many design decisions had to be made early. However, now that all of these experiments have been completed and the agents have been evaluated, we can examine areas where there might be room for improvement. We describe possibilities for future work here.

6.1.1 Test the New 3-Player Agents Against Humans

Soon after the CPRG created its first artificially intelligent Poker agents, those agents were played against human opposition. There are several advantages to testing new agents against humans. One advantage is that the skill level of humans varies greatly; this allows us to test against beginners and professionals alike. Another advantage is that feedback provided by more experienced players can provide us with better insight into the strengths and weaknesses of the agent. For instance, the CPRG has modified its card abstractions in the past based on human feedback. Finally, simply obtaining more data (e.g. hand histories and winrates) allows us to evaluate and compare new agents.

The CPRG generates many different heads-up limit Texas Hold'em agents with CFR. For simplicity, when any of these agents are played against humans, they are usually named *Polaris*. In 2007, *Polaris* played against professional Poker players, Phil "The Unibomber" Laak and Ali Eslami, in The First Man-Machine Poker Competition [7]. Even though *Polaris* lost the competition, it came very close and the Poker players were quite impressed. The next year, the 2008 Man-Machine Poker Competition was held in Las Vegas. This time around, a new-and-improved *Polaris* played against some of the best heads-up limit Texas Hold'em players in the world and won!

Now that two-player Poker agents have surpassed the human world-class level, it would be nice to attain the same feat for 3-player Poker. In the future, it would be both exciting and useful to test our new 3-player agents against humans (perhaps a master agent with improved HUEs as discussed in Section 6.1.2). However, there is a complication with the 3-player game that does not arise in two-player Hold'em. There are two configurations that could occur between our agents and humans: two agents and one human or two humans and one agent. When two humans are playing against one agent, we must ensure that the humans do not collude against the agent. One approach may be to place the two humans in separate rooms and anonymize the players' screen names so that they cannot determine which other player is the AI agent.

6.1.2 Improve the Performance of Heads-Up Experts

In Chapter 4, we saw that a master agent using HUEs provided an improvement over simply using the base player strategy *Multi2*. In fact, just using uniform strategy-seeded HUEs (i.e. assume players hold random hands) added to *Multi2*'s winrate! However, when the HUEs were used in conjunction with *IR16*, the master agent suffered a loss in winrate as compared to the base player.

It is encouraging that expert knowledge strategy-seeding out-performed uniform strategy-seeding for both base players, *Multi2* and *IR16*. This result demonstrates that the winrate is sensitive to the seeded strategy (i.e. *a priori* distribution of hole cards). In future work, we would like to create stronger expert-knowledge HUEs. In particular, we would like to seed HUEs with the preflop strategy of the base player that will be used with them. We feel, intuitively, that doing so would result in stronger play since the HUEs strategies would be much more consistent with the base player strategy.

In addition to improving HUEs with better strategy-seeding, we would also like to improve them via abstractions. In theory, refining an abstraction does not necessarily reduce the exploitability of a strategy profile [27]. However, in practice, using a more refined (i.e. larger) abstraction for sizable abstract Poker games results in a reduction of exploitability and a winrate improvement against a cross-table of opponents. All of the HUEs generated in this work have used a 5-bucket $E[HS^2]$ abstraction whereas abstractions several orders of magnitude larger have been used to generate heads-up strategies by the CPRG. The reason for using a 5-bucket abstraction was because HUEs were simply a proof of concept and, thus, we wanted to be able to generate their strategies quickly. Thus, in the future, we would like to create 8- or 10-bucket $E[HS^2]$ abstractions as we believe this would result in stronger HUEs.

6.1.3 Create New Abstractions for 3-player

Two abstractions were used for the 3-player agents described in this dissertation. *Multi2* uses a 2-bucket $E[HS^2]$ perfect recall abstraction and *IR16* uses a 16-bucket $E[HS^2]$ imperfect recall abstraction. However, hands were ranked based on their probability of winning against *one* random hand. There were several reasons for the decision to use this hand ranking. As we have seen in Chapter 4, between 50% and 70% of 3-player self-play hands resulted in a two-player subgame where one player folded preflop. By the time a showdown is reached, many more betting sequences will have only two players. Thus, using a hand ranking based on two-player hand strengths should still provide fairly sensible play.

One of the main reasons for using a two-player hand ranking instead of a 3-player hand ranking is the computational time required to generate the hand rankings. A two-player ranking can be generated in about a day or less. A 3-player *preflop* ranking generated by Ganzfried and Sandholm took a month on 16 processors [8]. Note that a *preflop* ranking (where all 5 community cards have been rolled out) takes less time than a ranking that includes hand strengths for all four rounds.

One possibility for a new hand ranking (given sufficient time to generate it) is a 3-player $E[HS^2]$ ranking where the expected hand strength squared of each hand is computed against *two* uniformly random hands. In this case, CFR should dynamically use a ranking corresponding to the number of active players. For example, use the 3-player $E[HS^2]$ ranking when no players have folded and the 2-player $E[HS^2]$ ranking when one player has folded.

Alternatively, we should test an abstraction which uses an estimate of the probability of winning against N opponents, $(E[HS])^N$ [1, p. 46]. This is an estimate since it makes the incorrect assumption that opponent hands are independent. It also ignores hand potential. However, in practice, this may provide a useful metric, $(E[HS])^2$, when there are 3 active players since the exponent is the number of opponents, not total players. The advantage to this estimated metric is that it still uses a two-player hand ranking meaning that it can be generated much faster than a 3-player hand ranking.

Bibliography

[1] D. Billings. *Algorithms and assessment in computer poker*. PhD thesis, Computing Science Department, University of Alberta, Edmonton, 2006.

[2] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 661–668, 2003.

[3] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134(1):201–240, 2002.

[4] D. Billings and M. Kan. A tool for the direct assessment of poker decisions. *The International Association of Computer Games Journal*, 29(3):119–142, 2006.

[5] M. Buro. The Othello match of the year: Takeshi Murakami vs. Logistello. *ICCA Journal*, 20(3):189–193, 1997.

[6] M. Buro, J.R. Long, T. Furtak, and N. Sturtevant. Improving state evaluation, inference, and search in trick-based card games. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI2009)*, 2009.

[7] The Second Man-Machine Poker Competition. http://poker.cs.ualberta.ca/man-machine/.

[8] S. Ganzfried and T. Sandholm. Computing an approximate jam/fold equilibrium for 3-player no-limit Texas Hold'em tournaments. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 919–925. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, 2008.

[9] S. Ganzfried and T. Sandholm. Computing Equilibria in Multiplayer Stochastic Games of Imperfect Information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[10] A. Gilpin and T. Sandholm. Optimal Rhode Island Hold'em poker. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 20, page 1684. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[11] A. Gilpin, T. Sandholm, and T.B. Sorensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 50. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[12] N. Grudzien and G. Herzog. *Winning in Tough Hold 'em Games: Short-Handed and High-Stakes Concepts and Theory for Limit Hold 'em*. Two Plus Two Publishing, 2007.

[13] A. Hodges. *Alan Turing: the enigma*. Walker & Company, 2000.

[14] F.H. Hsu. *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2002.

[15] M.B. Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, Computing Science Department, University of Alberta, Edmonton, 2007.

[16] D. Koller, N. Megiddo, and B. Von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259, 1996.

[17] H.W. Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, 1:97–103, 1950.

[18] J.F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, pages 48–49, 1950.

[19] M.J. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1994.

[20] J. Schaeffer, N. Burch, Y. Bjornsson, A. Kishimoto, M. Muller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *Science*, 317(5844):1518, 2007.

[21] J. Schaeffer, R. Lake, P. Lu, and M. Bryant. Chinook: The world man-machine checkers champion. *AI Magazine*, 17(1):21–29, 1996.

[22] B. Sheppard. World-championship-caliber Scrabble. *Artificial Intelligence*, 134(1-2):241–276, 2002.

[23] D. Sklansky. *The Theory of Poker*. Two Plus Two Publishing, 1999.

[24] F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and C. Rayner. Bayes-bluff: Opponent modelling in poker. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 550–558, 2005.

[25] N.R. Sturtevant. *Multi-player games: Algorithms and approaches*. PhD thesis, Computer Science Department, University of California, Los Angeles, 2003.

[26] G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[27] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron. Abstraction pathology in extensive games. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2009.

[28] M. Zinkevich, M. Bowling, and N. Burch. A new algorithm for generating strong strategies in massive zero-sum games. In *Proceedings of the Twenty-Seventh Conference on Artificial Intelligence (AAAI)*, 2007.

[29] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Neural Information Processing Systems*, volume 21, 2008.