A Practical Use of Imperfect Recall

Kevin Waugh, Martin Zinkevich†, Michael Johanson, Morgan Kan, David Schnizlein and **Michael Bowling**
{waugh, johanson, mkan, schnizle, bowling}@cs.ualberta.ca
maz@yahoo-inc.com†
Department of Computing Science
2-21 Athabasca Hall
University of Alberta
Edmonton, AB Canada T6G 2E8
Yahoo! Research†
2821 Mission College Blvd.
Santa Clara, CA, USA 95054

Abstract

Perfect recall is the common and natural assumption that an agent never forgets. As a consequence, the agent can always condition its choice of action on any prior observations. In this paper, we explore relaxing this assumption. We observe the negative impact this relaxation has on algorithms: some algorithms are no longer well-defined, while others lose their theoretical guarantees on the quality of a solution. Despite these disadvantages, we show that removing this restriction can provide considerable empirical advantages when modeling extremely large extensive games. In particular, it allows fine granularity of the most relevant observations without requiring decisions to be contingent on all past observations. In the domain of poker, this improvement enables new types of information to be used in the abstraction. By making use of imperfect recall and new types of information, our poker program was able to win the limit equilibrium event as well as the no-limit event at the 2008 AAAI Computer Poker Competition. We show experimental results to verify that our programs using imperfect recall are indeed stronger than their perfect recall counterparts.

Introduction

Perfect recall is the assumption that the rules of the game never require a player to forget her own past actions or any prior observations when making those actions. Kuhn 1953 first formalized the perfect recall assumption in a landmark work that showed the equivalence between behavioral strategies (where players randomize their strategies at choice points) and mixed strategies (where players randomize their strategies prior to playing) in any game exhibiting perfect recall. This equivalence allowed all of the theory of normal-form games to be applied to extensive games with perfect recall. For the next forty years, imperfect recall games were relegated to "awkward exceptions" (Ambrus-Lakatos 1999). Piccione and Rubinstein 1996 sparked a revival of interest in imperfect recall with their *paradox of the absent-minded driver*. This initial work and the resulting flurry of responses (*e.g.*, (Gilboa 1997; Piccione & Rubinstein 1997; Ambrus-Lakatos 1999)) focused mostly on the interpretation of imperfect recall: when and how players make decisions and with what knowledge. These works also showed how strange behaviour can arise in certain games of imperfect recall.

In this paper, we examine the computational, rather than philosophical, implications of imperfect recall. From the perspective of artificial intelligence, imperfect recall is more than just a modeling choice to describe a strategic interaction. Imperfect recall can be used to limit the space and size of strategies under consideration with the goal of reducing the computational burden of constructing an effective strategy. Consider a perfect recall strategy for heads-up Texas Hold'em. This game has 10^{18} game states with each player having 10^{14} information sets, requiring petabytes of disk space to even store a strategy. To compute effective strategies in games of this size, typically one first employs an abstraction technique (Billings *et al.* 2003; Gilpin & Sandholm 2006; 2007) to create a much smaller game. To date, only abstractions that preserve perfect recall have been considered, but the ability to forget allows more flexibility when designing an abstract game. An abstraction can allow more granularity of information at early decisions if this information does not need to be recalled at every later decision point.

Unfortunately, relaxing the perfect recall assumption results in the loss of most of the useful theoretical properties that resulted from Kuhn's work. A variety of issues arise with the efficient algorithms for finding equilibria without this assumption. Algorithms based on sequence form representations cease to be well-defined. Regret-based algorithms, while remaining well-defined for a large class of imperfect recall games, apparently lose their theoretical guarantees. Despite the loss of these guarantees, we forge ahead using a variant of the counterfactual regret algorithm (Zinkevich *et al.* 2008) to construct strategies based on imperfect recall abstractions in two intractable variants of poker. We show that the conceptual advantages of imperfect recall hold in practice and, despite the theoretical problems, the resulting strategies outperform their perfect recall counterparts.

Background

An extensive game is a useful tool for modeling how multiple agents interact with an environment. At each step a player or chance takes an action as the game progresses towards a terminal history. At a terminal history, players are rewarded or penalized based on the terminal that was

reached. To incorporate imperfect information, not all actions are fully observable to each player. This results in certain histories being indistinguishable to a player when she is faced with a decision.

Definition 1 (Extensive Game) *(Osborne & Rubenstein 1994, p. 200) A finite extensive game with imperfect information, Γ, 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**. No action can be taken from a terminal history and hence a terminal history is not a prefix of any other history. $A(h) = \{a : h \circ a \in H\}$ are the actions available after a non-terminal history $h \in H \setminus Z$.*
- *A **player function** P that assigns to each non-terminal history a member of $N \cup \{c\}$, where c represents chance. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$, then chance determines the action taken after history h. Let H_i be the set of histories where player i chooses the next action.*
- *A function f_c that associates with every history $h \in H_c$ a probability measure $f_c(\cdot|h)$ on $A(h)$. $f_c(a|h)$ is the probability that a occurs given history h is reached, where each such probability measure is independent of every other such measure.*
- *For each player $i \in N$, a partition $\mathbf{I}_i$ of H_i with the property that $A(h) = A(h') = A(I)$ whenever h and h' are in the same member of the partition, I. $\mathbf{I}_i$ is the **information partition** of player i; a set $I \in \mathbf{I}_i$ is an **information set** of player i.*
- *For each player $i \in N$, a **utility function** u_i that assigns each terminal history a real value. $u_i(z)$ is rewarded to player i for reaching terminal history z. If $N = \{1, 2\}$ and for all z, $u_1(z) = -u_2(z)$, an extensive form game is said to be **zero-sum**.*

Two histories belonging to the same information set are indistinguishable to the acting player. Thus, the player cannot condition her choice of action on anything other than the information set that contains that history. This can lead to awkward and unnatural games where a player is forced to forget (*i.e.*, not be able to condition her action on) information that she previously knew. Games that display this behaviour are thought of as oddities, unnecessarily difficult, and usually dismissed. Typically, **perfect recall** is assumed, which is a condition on the information partitions to exclude these situations. A game exhibits perfect recall if from any information set a player can determine her own past information sets as well as the action taken from those information sets. This condition is satisfied only when all histories in an information set share the same past information sets and same past actions for the acting player. A game is said to exhibit **imperfect recall** if this condition does not hold.

When playing an extensive game, we call the mechanism that a player uses to make her decisions a strategy. Similarly, we call the combination of all player's strategies a strategy profile.

Definition 2 (Strategy) *We call $\sigma_i \in \Sigma_i$ a **strategy for player i**. $\sigma_i(\cdot|I)$ defines a probability distribution on $A(I)$ for all $I \in \mathbf{I}_i$. Upon reaching a history in I, player i samples an action from $\sigma_i(\cdot|I)$ and then plays the sampled action.*

Definition 3 (Strategy Profile) *We call $\sigma \in \Sigma$ a **strategy profile**. It contains one strategy for each player. We denote σ_{-i} as the profile containing all strategies except for player i's. We define $u_i(\sigma)$ as the expected utility of player i given that all players play according to σ.*

A natural solution concept for an extensive game is the Nash Equilibrium. A strategy profile is at equilibrium if no player can benefit by deviating his or her strategy from the one given in the profile. A strategy profile is said to be near equilibrium if any player's incentive to deviate is marginal.

Definition 4 (Equilibrium) *A **Nash Equilibrium** is a strategy profile, σ, such that for all $i \in N, \sigma'_i \in \Sigma_i$:*

$$u_i(\sigma) \geq u_i(\sigma_{-i} \cup \sigma'_i) \tag{1}$$

*An **ε-Nash Equilibrium** is a strategy profile σ such that for all $i \in N$ and $\sigma'_i \in \Sigma_i$:*

$$u_i(\sigma) + \varepsilon \geq u_i(\sigma_{-i} \cup \sigma'_i) \tag{2}$$

For zero-sum games, there exists efficient procedures for computing ε-equilibrium profiles, such as linear programming using sequence form (Koller, Megiddo, & Stengel 1996), counterfactual regret minimization (Zinkevich *et al.* 2008) and gradient-based methods (Gilpin *et al.* 2007). In a zero-sum game, playing a strategy belonging to an equilibrium profile maximizes a player's worst-case expected utility.

Chance Sampled Counterfactual Regret Minimization

One efficient algorithm for computing a ε-equilibrium in a zero-sum game with perfect recall is the chance sampled counterfactual regret minimization algorithm. This algorithm is quite easy to implement, and with high probability will converge to an equilibrium profile as the number of iterations increases. The algorithm is detailed more fully in Zinkevich *et al.* 2008, but we shall review it here for completeness.

First, a few definitions. We let $\pi^\sigma(h)$ be the probability that history h is reached given that all players play according to σ. Similarly, we can define $\pi^\sigma_i(h)$ as the the portion of $\pi^\sigma(h)$ resulting from player i's actions and $\pi^\sigma_{-i}(h)$ as the portion resulting from the actions of all players (and chance) except for player i. Similar constructs of the form $\pi^\sigma_*(h, h')$ are defined as $*$'s contribution to the probability of reaching h' given that h is reached. Given these definitions, we let $\pi^\sigma_*(I) = \sum_{h \in I} \pi^\sigma_*(h)$ be $*$'s contribution to the probability of reaching information set I. We let $u_i(\sigma, I)$ be the counterfactual utility for player i at information set I. That is, $u_i(\sigma, I)$ is the expected utility for player i given that information set I is reached and all players play according to σ afterwards. Mathematically, we have $u_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi^\sigma_{-i}(h)\pi^\sigma(h, z)u_i(z)$. One final bit of notation we will need is that $\sigma|_{I \to a}$ denotes the strategy

profile where at information set I action a is chosen and at all other information sets the action is chosen according to σ.

As we proceed through iterations of the algorithm, we will have to keep track of some information. The first of which we denote σ^T, the current strategy profile at time T. We set σ^0 to be an arbitrary strategy profile. For each iteration starting with $T = 1$, we will use the previous iteration's strategy profile, along with some accumulated regret information, to compute a new strategy profile. The average of all these strategy profiles, $\bar{\sigma}^T$, is also maintained. Ultimately, it is $\bar{\sigma}^T$ that converges to a ε-equilibrium. The regret information we need is $R_i^T(I, a)$, which denotes the counterfactual regret up to time T on action a at information set I experienced by player i. That is, this quantity is how much counterfactual utility player i would have gained from only playing action a at information set I, as opposed to playing her regret minimizing strategy, had she played to reach information set I and her opponent played according to the most recent strategy profile. Initially, we set $R_i^0(I, a) = 0$ for all information sets and actions.

On each iteration, we first update the counterfactual regret information and then compute a new strategy profile using the updated regret totals. The counterfactual regret is updated using the following formula:

$$R_i^T(I, a) = R_i^{T-1}(I, a) + u_i(\sigma^{T-1}|_{I \to a}) - u_i(\sigma^{T-1}, I) \tag{3}$$

We use the well-known regret matching equation, which relies on Blackwell's approachability theorem, to update the strategy profile as follows:

$$\sigma^T(a|I) = \frac{\max\{0, R_i^T(I, a)\}}{\sum_{a \in A(I)} \max\{0, R_i^T(I, a)\}} \tag{4}$$

This update procedure ensures that the counter-factual regret at each information set decreases to zero. As these regret terms bound the overall regret, it too approaches zero as the number of completed iterations increases.

What is described above is the standard couterfactual regret minimizing algorithm. To convert this algorithm to the chance sampled variant, all we must do is randomly sample chance's strategy at each iteration. All the probabilities in the updates are then replaced with the corresponding probabilities where chance plays according to the sampled strategy. Given this change, the average strategy profile approaches an equilibrium with high probability. This change can drastically effect the performance of the algorithm as in certain games, such as many poker variants, the computation required on for iteration is dramatically simplified.

Motivation for Imperfect Recall

Many games of interest to the artificial intelligence community, though exhibiting perfect recall, are far too large to feasibly compute an equilibrium profile. As noted in the introduction, two-player limit Texas Hold'em has approximately 10^{18} game states and would require petabytes of memory to record a strategy. In two-player no-limit Texas Hold'em, there are many more actions available to the players, increasing the number of game states to approximately 10^{71}. State-of-the-art techniques for finding equilibria cannot handle games of this size. In order to make use of game theoretic approaches for computing strategies in these games, we must make use of abstraction techniques (Billings *et al.* 2003; Gilpin & Sandholm 2007). These approaches create a smaller abstract game that we hope accurately models the original game. An abstraction technique reduces the amount of information available to a player at a decision point. Commonly, this is done by further obscuring chance's actions, *i.e.*, some of chance's actions that in the original game are distinguishable to a player are grouped together so that they no longer are distinguishable in the abstract game. Once the abstract game is created, we can then use modern techniques to solve for an ε-equilibrium in this smaller game and use the resulting strategies to play the original game. The hope is that the error introduced by abstraction is not too large and therefore the induced strategy for the original game is of suitable quality. Prior to this work, the smaller abstract games have always exhibited perfect recall.

Although exclusively used, perfect recall can be troublesome when creating abstract games. Early in the game, an agent may be forced to have inadequate information to make an informed decision because the agent would have to remember the information for the remainder of the game. Often including enough information in the abstract game to properly make these decision would increase the size of the abstract game beyond what can be solved. Later in the game, much of the information available to the player is what has been remembered from past actions. Some of the past information may still be relevant, but often is it less important than the most recent information. Here, the less relevant information is in a sense taking the space of information that could be more useful in making a decision. We can visualize these problems in Figure 1. Here, the information available to a player (shown horizontally) on consecutive rounds (shown vertically) is represented as the sum of the player's past actions (denoted P) and as chance's abstracted actions (denoted 1, 2 and 3). The bulk of the strategy space in many games is occupied by decisions made late in the game, which is after chance has taken multiple actions. Since this space is limited, we must appropriately size chance's initial actions as they are remembered through the entire game.

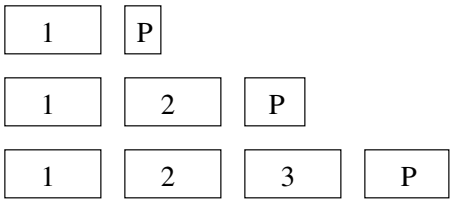

Figure 1: Information in an Abstract Perfect Recall Game

Using imperfect recall when creating abstract games allows us to alleviate these problems to some degree. At a decision, we can focus the information available to an agent on the most relevant information. At later decisions, we can either choose to forget past information (which was once relevant) or modify its granularity to what is deemed an accept-

able level. This allows us more flexibility in choosing an abstract game. Additionally, it allows us to take further advantage of domain knowledge and provide what is believed to be the most relevant information to the agent when it makes its decision. We see this contrast visually in Figure 2.

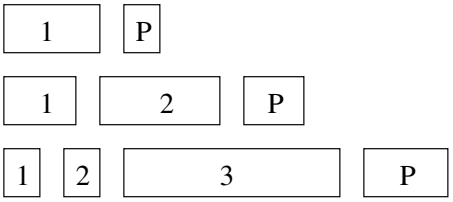

Figure 2: Information in an Abstract Imperfect Recall Game

Challenges of Imperfect Recall

Though imperfect recall seems advantageous from a modeling standpoint, many computational issues arise when faced with games of imperfect recall.

Conceptual Challenges. Consider the zero-sum game in Figure 3. In this two player game, the first player chooses a direction initially, left or right, and tells this direction to the second player. The second player then decides whether she wishes to continue playing the game, or to abstain from playing. Abstaining from play results in her receiving a penalty. If she decides to proceed, her memory is erased of the direction chosen by the first player. She must then repeat which direction was picked in the beginning. Answering this question correctly gives her a small reward, where answering incorrectly is penalized heavily. There are two simple strategies in this game where she will never answer incorrectly. The first is to abstain always when left is picked and to play and answer right otherwise. The second is the symmetric strategy where the player abstains when right is picked. Interestingly, if she is rational and privileged to the first player's strategy, she will always pick one of these two strategies to maximize her reward. Furthermore, she will never randomize her strategy after deciding to play as the penalty for answering incorrectly is too large. As a consequence, she cannot guarantee a reward of more than -1. A strategy by the first player that randomizes between left and right with equal probability guarantees a reward of $1/2$. This is the maximum reward that the first player can guarantee as any bias towards either side will result in the second player choosing the pure strategy that correctly guesses that biased direction. We note here that there is a gap in the rewards, and this is a consequence of the fact that there is no equilibrium in this game[1]. We should note that our goal is not to solve imperfect recall games, as we cannot hope to achieve this without the concept of an equilibrium, but instead to efficiently find good strategies for large perfect recall games.

[1]This does not contradict Nash's important result that every game has a mixed strategy equilibria as we are looking at behavioral strategies which are not necessarily equivalent in imperfect recall scenarios.

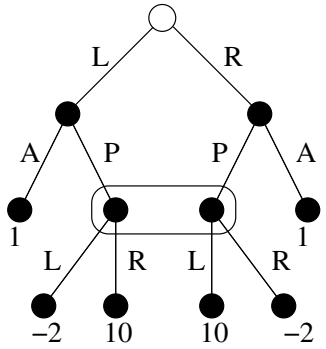

Figure 3: An example of a game with imperfect recall

With this goal in mind, the potential lack of an equilibrium in our abstract games is discouraging, but does not halt the idea completely.

Algorithmic Challenges. One method for finding good strategies in an imperfect recall game is to convert the game into one of perfect recall. This can be accomplished with the notion of *multiple selves* (Gilboa 1997). Each player with imperfect recall is replaced with multiple players, each with the same utility function. These extra players can then be privileged to different information so no actual player is forgetting any of their past actions or decisions. Unfortunately, we do not have efficient techniques for solving n-player games, with $n > 2$, even when they exhibit perfect recall. That is, additional players beyond two, and non-constant-sum payoffs have their own set of equally difficult challenges.

Another direct approach is to attempt to solve the imperfect recall game explicitly. Koller and Megiddo 1996 presented an algorithm for just this, but it has two issues that make it impractical in our situation. First, the algorithm requires exponential time to complete. Second, the resulting strategy is in a different space, one that requires exponential size to store. They showed in a previous work that solving an imperfect recall game is indeed NP-hard (Koller & Megiddo 1992).

Many techniques for solving zero-sum games make use of **sequence form**. Sequence form makes use of a **realization plan**, which is a linear representation of a strategy in a game of perfect recall. Using this linear representation, one can construct a linear program similar to the one used to solve for equilibria matrix games. This linear program can be solved directly (Koller, Megiddo, & Stengel 1996), but large scale methods can exploit the structure of this problem and use specialized gradient-based methods to converge more rapidly and use fewer resources than a standard linear program solver (Gilpin *et al.* 2007). Unfortunately, the very definition of a realization plan relies on the fact that a single action from an information set uniquely defines an entire sequence of actions under perfect recall. This no longer holds when perfect recall is relaxed. As the definition of a realization plan is not well-defined under imperfect recall, al-

gorithms based on sequence form are themselves ill-defined when perfect recall is omitted.

As we reviewed in the background, the notion of counterfactual regret (Zinkevich *et al.* 2008) is used to extend the concept of regret to extensive games with perfect recall. It is well known that if two agents use regret minimizing strategies to compete in repeated play of a zero-sum game that their average strategies converge to an equilibrium profile. Here, averaging a strategy refers to averaging the probability distribution at each information set where each distribution is weighted by the probability that the underlying strategy will reach that information set. An important property of averaging a strategy is that under perfect recall this averaging operation is linear in regard to the expected utility of a player. That is, if there are n strategies for the first player, then for any strategy for the second player, the average expected utility of the n strategies is the same as the expected utility of the average of the n strategies. This clearly does not hold in the example game in Figure 3 when we average the two pure strategies for the second player. Unfortunately, the proof of convergence to an equilibria hinges on this fact. As previously noted, the concept of counterfactual regret is defined as the regret at an information set in terms of counterfactual utility. Conceptually, the counterfactual utility at an information set is concerned with how a player chooses her actions to try to reach said information set. With certain imperfect recall games, the notion of trying to reach an information set becomes dubious and the notion of counterfactual regret becomes ill-defined. For example, if from an information set a player can take two separate actions that both can lead to the same future information set, then which action should the player choose to try to reach that future information set? If we impose a more strict condition than imperfect recall, where no player cannot reach the same future information set through separate actions from a past information set, this ambiguity is resolved. One further restriction we must impose is that no play of the game may visit the same information set twice. With the chance sampled variant of CFR, once we have sampled chance's actions all the operations performed on a single iteration behave exactly the same for a game from this new class as they would on a game of perfect recall. That is, we do not have to modify our chance sampled algorithm to account for imperfect recall for it to be well-defined, but we will lose the guarantee of approaching an equilibrium should we provide a game from this new class that does not exhibit perfect recall.

Imperfect Recall Abstraction in Poker

For the remainder of the paper, we explore the use of imperfect recall abstractions in the domain of poker. We use CFR to find strategies for the resulting abstract games and compare them to perfect recall counterparts. As a test domain, we use two variants of heads-up Texas Hold'em, which are zero-sum poker games. This allows us to compare our new programs with prior entries to the AAAI Computer Poker Competition. In this section, we will first briefly describe the Texas Hold'em variants. We must then describe previous abstraction techniques as well as our new imperfect recall abstraction techniques before we compare our new programs to previous programs that make use of perfect recall abstractions.

Texas Hold'em

Texas Hold'em games require a standard deck of cards, which is shuffled prior to play. One player is designated the *small blind* and one the *big blind*. This designation typically alternates on every hand. Before being dealt any cards, the small blind is forced to bet one chip and the big blind two chips into the *pot*. After these forced bets, four rounds of play occur. In each round, some cards are dealt from the top of the deck and subsequently players get to bet. The rules for how players are allowed to bet depends on the type of Texas Hold'em game. The two variants we are concerned with in this paper are *limit* and *no-limit*. Limit betting is assumed unless otherwise specified. The first round is called the *preflop* and consists of two private cards being dealt to each player. The small blind starts the betting during the preflop. The preflop is followed by the *flop*, where three community cards are dealt face up. The *turn* and the *river* follow the flop. One community card is dealt during each of these rounds. The big blind starts the betting for the flop, turn and river. After the river betting has completed, players make the best five card poker hand from their two private cards and the five community cards. The player with the best hand wins all the chips in the pot.

During the betting portion of a round, the players alternate making betting decisions. When facing a bet, *i.e.*, the opposing player has more chips in the pot than the player to act, a player may *fold*, *call* or *raise*. Folding immediately ends the game and forfeits all chips in the pot to the opposing player. Calling requires the player to match the opposing player's bet. Raising requires a player to exceed the opposing player's bet. When not facing a bet, a player can *check*, where no additional chips are added to the pot, or raise. If checking or calling is the first action of a round then action moves to the opponent, otherwise the game proceeds to the next round. In a limit game, the size and number of raises is fixed. In particular, the preflop and flop have a raise size of two chips and the turn and river have a raise size of four chips. The preflop has a maximum of three raises per round and all subsequent rounds have a maximum of four raises per round. In a no-limit game, a player may bet any number of chips in his remaining stack provided that the raise is either at least as big as the most recent raise for that round or it puts the player *all-in*. Here, raising all-in refers to betting all of one's remaining chips. In our no-limit game, each player starts each game with one thousand chips.

Abstraction

As the poker games we are interested in are far too large to solve directly, we employ the use of abstraction techniques to create smaller games that can be solved directly.

For both limit and no-limit games, we must perform card abstraction. In the abstract game, a player knows that the hand it holds belongs to a particular set of hands, as opposed to an exact hand. This in effect merges information sets together. Various different metrics have been used in the past to create these hand groupings. The most successful metrics

incorporate some notion of *strength*, which is how likely a hand will win once all cards have been dealt, and *potential*, which is how likely a hand's strength will improve or diminish as future cards are dealt. We use *hand strength squared* as our metric for grouping hands, which incorporates both of these good qualities.

In no-limit games, we must perform action abstraction in addition to card abstraction. Action abstraction restricts the type of actions a player can make. That is, in a no-limit game, we disallow certain bet sizes to reduce the size of the game. Typically, to play the original game there must be a translation mechanism (Gilpin, Sandholm, & Sorensen 2008) to convert actions in the original game to ones available in the abstract game. The sizes we allow for raises are a pot sized bet, a ten pot sized bet and the all-in bet. Since all our programs play with the same betting abstraction, translation is irrelevant for these experiments.

Public Information

Previous abstraction techniques would only provide the agent with information regarding the strength of its hand. This information does not differentiate whether the strength of a hand is a result of the community cards or of the agent's private cards. This differentiation is strategically important. For example, on a *dry* board, which is one where it is unlikely that either player has a strong hand, a player should not bluff as often as on other types of boards. An observant opponent will quickly realize that often the player does not often have a strong hand in this situation. Similarly, on a *connected* board, which is one where it is likely a player has either made a strong hand or is drawing to a strong hand, a player might be less aggressive in betting his or her strong hands as it is more likely that an opponent also has a strong hand. In this situation, looking at the absolute hand strength is deceiving as a hand can be weak relative to likely opponent holdings and still have a high absolute strength. Some public information can be derived by an agent by looking at the history of hand strengths through the betting rounds, but there still exists important situations that remain indistinguishable.

Our new programs makes additional use of the community cards on the flop and the turn. As it is still not possible for our program to differentiate every single board, we cluster boards into similar categories. To create our board clusters we make use of a perfect recall abstraction with 10 buckets per round. In this abstraction, that each time chance acts, its actions are uniformly divided into 10 different groups based on the hand strength squared metric. Using this perfect recall abstraction, we create a 10 by 10 transition table for every possible set of community cards, where an entry (i, j) in this table denotes the number of hands that prior to chance acting where in bucket i that after chance's action ended up in bucket j. We then run K-Means clustering using the Euclidean distance metric for 10000 iterations. Our program uses 20 public information buckets on the flop and these buckets are further divided into 3 additional buckets on the turn.

Results

All of our programs were trained using the chance sampled counterfactual regret minimization algorithm. The number of iterations used to compute the strategies was between 500 million for the smaller abstract games, to 10 billion for the larger abstract games. The smaller games took about a day of computation on 8 nodes of a powerful cluster to compute, where as the larger abstract games required the same resources for about a week. We used millibets per hand (mb/h) as our unit of measurement when comparing two strategies, which is one thousandth of a small bet. That is, if one program beats another by 5 millibets per hand, it is expected to win 1 cent from the other player per hand (when playing with a 2 dollar big blind). Each of the programs were played against each other in 10000 hand duplicate matches until the 95% confidence interval was no larger than ± 2 millibets in limit and no larger than ± 64 millibets in no-limit.

In Table 1 we see the results of a tournament between eight different limit players. The first three bots use an *8s* sized card abstraction, which has 23 million information sets. The first of these programs, *pr.8*, uses a perfect recall abstraction. The second, *ir.preflop.8*, can distinguish all 169 preflop hands, but forgets all of this information on the flop. On the flop, all hands a uniformly grouped into 64 buckets. These flop buckets are remembered for the remainder of the game. This abstraction is essentially the same size as *pr.8* as it contains only 161 more information sets. The third program, *ir.8*, forgets its past buckets on every round and instead uses all of its memory for the finest granularity on current hand strength. That is, all hands are grouped into 64 buckets on the flop, 512 buckets on the turn and 4096 buckets on the river. This program has perfect information preflop. The next two programs in the table are perfect recall programs using *12s* and *14s* sized abstractions respectively. The 12s abstraction has 118 million information sets and the 14s abstraction has 219 million information sets. Finally, our last three bots make use of new public information. The first of these programs uses an approximately 12s sized abstraction and perfect recall from the flop onward. It uses perfect information preflop. On the flop, its hands are grouped into buckets based on the 20 public information buckets and 8 hand strength buckets. These flop buckets are remembered for the remainder of the game. The turn and river have 12 hand strength buckets each. The second of these programs has a higher granularity of hand strength information on the flop, but it reduces this granularity for future rounds. That is, it has 16 hand strength buckets on the flop, but on the turn and river, the program only recalls the flop hand strength as if there were only 8 buckets available. These additional hand strength buckets on the flop do not drastically impact the size of the strategy as they are not remembered on future rounds. The third program balances hand strength information and public information by reducing the granularity of past hand strength information as the game progresses. This last program is approximately 14s sized and additionally has public information on the turn.

We see in the limit game that imperfect recall alone does not appear to provide a significant improvement in play. The 8s sized players perform similarly against all other players

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) pr.8	0	-1	1	-14	-18	-14	-15	-18
(2) ir.preflop.8	-1	0	0	-13	-16	-13	-13	-18
(3) ir.8	1	0	0	-10	-14	-9	-10	-15
(4) pr.12	14	13	10	0	-5	-5	-5	-10
(5) pr.14	18	16	14	5	0	0	-1	-7
(6) flop.12	14	13	9	5	0	0	-2	-7
(7) flop.12-2	15	13	10	5	1	2	0	-6
(8) flop.turn.14	18	18	15	10	7	7	6	0

Table 1: Heads-up Texas Hold'em Crosstable in millibets per hand (mb/h)

in the tournament and tie each other. The imperfect recall 8s players lose sightly less than the perfect recall 8s player against the remainder of the field.

The power of imperfect recall in limit appears with the addition of public information. The 12s sized players with public information perform better than the perfect recall 12s player and perform similarly to the 14s sized player, which is approximately two times larger. The second of the 12s sized imperfect recall players actually beats the 14s sized perfect recall player. The 14s sized imperfect recall player, which we expected to be the strongest, performs about on par with the 14s sized perfect recall player against the 8s sized programs, but performs much better against the larger programs.

The *flop.turn.14* player was entered into the 2008 AAAI Computer Poker Competition limit events. It won the limit equilibrium event by beating all other competitors with statistical significance.

In Table 2 we see the results of a tournament between five different no-limit players. Three of these players play in an *8s* sized abstraction, and two of these players play in a *12s* sized abstraction. The *ir.preflop* players make use imperfect recall to see all possible preflop situations. These players forget all information about what they held on the preflop when the flop has been reached. The *ir.8* player uses imperfect recall on every round, which gives it the finest granularity of the player's current hand strength, but no memory of any past hand strengths.

The player that performs the worst is the 8s player using perfect recall. Somewhat surprising, however, is that the perfect recall 12s player is worse than the imperfect recall 8s players. This player beats the perfect recall 8s player by 435 mb/h, the largest amount in the table, but loses by moderate amounts to all the other players. This is especially important to note, as the size of the strategy the 12s player uses is roughly five times larger than that of the 8s players.

We observe that the 8s player that uses imperfect recall on every round beats every other player except the imperfect recall 12s player. This is slightly different from the results in limit, where imperfect recall alone does not seem to have much of an effect. A possible explanation for this is due to the presence of the all-in bet in no-limit. When facing an all-in bet, a very important consideration of the acting player is the strength of his or her hand. The imperfect recall players have the highest granularity on this particular information. In limit, one individual decision is of less importance to ones overall strategy. In particular, the preflop decisions in limit are much easier to correct for later in the game, whereas in no-limit it can be extremely costly to bet a large amount of one's chips preflop with a mediocre hand.

Finally, we see that the imperfect recall 12s player beat every other player, including the perfect recall 12s player by 173 mb/h. What is interesting to note, is that it only beats the perfect recall 8s player by 386 mb/h, less than the 435 mb/h the perfect recall 12s player accomplishes. This means that using imperfect recall is not a strict benefit in all situations.

The *ir.preflop.12* player was entered into the 2008 AAAI Computer Competition no-limit event. It won the event, which was determined using a bankroll runoff system. In this system, all players play each other in a round-robin tournament. The player that loses the most to all other players is then eliminated and the chips it lost are removed from the other players' totals. This process is repeated to determine the place of all players.

Conclusion

Perfect recall is a common assumption for extensive games and for building abstractions — and for good reason. Imperfect recall creates numerous conceptual and algorithmic difficulties, ranging from the loss of the usual solution concept to certain algorithms no longer even being well-defined. From the artificial intelligence perspective, though, abandoning the perfect recall assumption allows for far more control in constructing abstractions that give players the most relevant information for the available computational resources. Although without theoretical guarantees, we showed how we can use imperfect recall abstractions to build strong strategies in two varieties of poker domains. We demonstrated the superiority of the imperfect recall strategies over their perfect recall counterparts.

Acknowledgments

The authors of this paper would like to thank the current and former members of the Computer Poker Research Group at the University of Alberta for helpful conversations leading to this work.

References

Ambrus-Lakatos, L. 1999. An essay on decision theory with imperfect recall. IEHAS Discussion Papers 9905, Institute of Economics, Hungarian Academy of Sciences.

	(1)	(2)	(3)	(4)	(5)
(1) pr.8	0	-252	-327	-435	-386
(2) ir.preflop.8	252	0	-47	68	-135
(3) ir.8	327	47	0	134	-56
(4) pr.12	435	-68	-134	0	-173
(5) ir.preflop.12	386	135	56	173	0

Table 2: Heads-up no-limit Texas Hold'em Crosstable in millibets per hand (mb/h)

Billings, D.; Burch, N.; Davidson, A.; Holte, R.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*.

Gilboa, I. 1997. A comment on the absent-minded driver paradox. *Games and Economic Behavior* 20(1):25–30.

Gilpin, A., and Sandholm, T. 2006. A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Gilpin, A., and Sandholm, T. 2007. 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 (AAAI)*. AAAI Press.

Gilpin, A.; Hoda, S.; Peña, J.; and Sandholm, T. 2007. Gradient-based algorithms for finding nash equilibria in extensive form games. In *Proceedings of the Eighteenth International Conference on Game Theory*.

Gilpin, A.; Sandholm, T.; and Sorensen, T. B. 2008. A heads-up no-limit texas hold'em poker player: discretized betting models and automatically generated equilibrium-finding programs. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*.

Koller, D., and Megiddo, N. 1992. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior* 4:528—552.

Koller, D., and Megiddo, N. 1996. Finding mixed strategies with small supports in extensive games. *International Journal of Game Theory* 25:73–92.

Koller, D.; Megiddo, N.; and Stengel, B. V. 1996. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior* 14:247–259.

Kuhn, H. 1953. *Contributions to the Theory of Games*, volume 2. Princeton University Press.

Osborne, M., and Rubenstein, A. 1994. *A Course in Game Theory*. The MIT Press.

Piccione, M., and Rubinstein, A. 1996. The absent minded driver's paradox: Synthesis and responses. Papers 39-96, Tel Aviv.

Piccione, M., and Rubinstein, A. 1997. On the interpretation of decision problems with imperfect recall. *Games and Economic Behavior* 20(1):3–24.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS)*.