Abstraction Pathologies in Extensive Games

Kevin Waugh, David Schnizlein, Michael Bowling, Duane Szafron
{waugh, schnizle, bowling, duane}@cs.ualberta.ca

Department of Computing Science
University of Alberta
Edmonton, AB, Canada T6G 2E8

ABSTRACT

Extensive games can be used to model many scenarios in which multiple agents interact with an environment. There has been considerable recent research on finding strong strategies in very large, zero-sum extensive games. The standard approach in such work is to employ abstraction techniques to derive a more tractably sized game. An extensive game solver is then employed to compute an equilibrium in that abstract game, and the resulting strategy is presumed to be strong in the full game. Progress in this line of research has focused on solving larger abstract games, which more closely resemble the full game. However, there is an underlying assumption that by abstracting less, and solving a larger game, an agent will have a stronger strategy in the full game. In this work we show that this assumption is not true in general. Refining an abstraction can actually lead to a weaker strategy. We show examples of these abstraction pathologies in a small game of poker that can be analyzed exactly. These examples show that pathologies arise when abstracting both chance nodes as well as a player's actions. In summary, this paper shows that the standard approach to finding strong strategies for large extensive games rests on shaky ground.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**] Problem Solving, Control Methods, and Search

General Terms

Algorithms, Economics, Reliability

Keywords

Abstraction, Game Theory, Equilibrium, Pathologies

1. INTRODUCTION

Extensive games provide a general model of strategic interaction between multiple agents. They can be used to describe sequential decision-making scenarios even in settings of imperfect information. They are a generalization of finite-horizon Markov decision processes to multiple agents, normal-form games to sequential actions, and standard search game trees (*e.g.*, games like go and chess) to imperfect information. As such they have the capacity to model a broad range of interesting strategic decision-making scenarios.

Extensive games have received considerable attention recently as the natural model for decision-making in poker. In fact, this research on artificial intelligence in poker has driven substantial algorithmic advancements in solving zero-sum extensive games. The traditional linear programming technique of Koller and Pfeffer [8] was unrivaled for over a decade, and could be used to solve games with 10^8 game states. In the past three years, a number of competing techniques have been developed by competitors in the AAAI Computer Poker Competitions [15], with some now able to solve games with about 10^{12} game states [14, 3]. Work has also started on games with more than two players [2].

While algorithmic advances have made it possible to solve very large extensive games, the smallest variants of poker still have many more states (two-player, limit Texas Hold'em has approximately 10^{18} game states [1]). The standard approach to this problem is to use abstraction [1]. The idea is to construct a tractably sized game that abstracts (less important) details of the players' information at each decision. This game is constructed to be small enough that a near equilibrium solution can be found tractably, and this solution will hopefully perform well in the full game. The algorithmic advances in solving large extensive games allow for finer and finer abstractions. These finer abstractions result in bigger abstract games, and presumably stronger strategies than the strategies found using coarser abstractions.

In this paper, we examine the effect of abstraction on the strength of the resulting abstract game solution. In particular, we examine the key presumption of the previous paragraph that finer abstractions will make the resulting strategy stronger in the full game. Our conclusion is that this presumption is not true in general. We, in fact, show a variety of *abstraction pathologies*: examples where refining an abstraction can result in a weaker strategy in the full game. These counterexamples are all in a small poker variant with only six cards, which allows us to compute exact measurements of a strategy's strength. The pathologies are also not limited to abstractions involving only chance nodes (common to all poker abstractions) or only players' actions (common to abstractions in no-limit variants).

2. BACKGROUND

We begin by formalizing the concepts of an extensive game, the Nash equilibrium solution concept, and abstraction.

Cite as: Title, Author(s), *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

2.1 Extensive Games

Extensive games are a useful tool for modeling environments with multiple agents. Players (and chance) alternate taking actions until a terminal history is reached, and the resulting utility to each of the players is determined by this final history. However, players may not be able to completely observe the actual choices of the other players or chance. This allows for scenarios where two distinct histories of actions may not be distinguishable by the next player to act, creating a game of imperfect information.

Formally, we can define an extensive game as follows.

DEFINITION 1 (EXTENSIVE GAME). *[10, p. 200] A finite extensive game with imperfect information is denoted* Γ *and 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*.* $Z \subseteq H$ *is the set of* ***terminal histories****.* $A(h) = \{a : (h, a) \in H\}$ *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 *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_i *called the* ***information partition*** *of player* i*; a set* $I_i \in \mathcal{I}_i$ *is an* ***information set*** *of player* i*. For every* $h, h' \in I_i$*, we require that* $A(h) = A(h')$*.*
- *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****.*

Histories in the same information set are indistinguishable to the player making the decision. This results in some information partitions that force odd and unrealistic situations on a player where they are forced to forget their own past histories or decisions. If all players can recall their previous actions with the corresponding information sets, the game is said to be one of **perfect recall**. This work will focus on finite, zero-sum extensive games with perfect recall.

A **strategy for player** i, σ_i, in an extensive game, Γ, is a function that assigns a probability distribution over $A(h)$ to each $h \in H_i$, where $\sigma_i(h) = \sigma_i(h')$ for all h and h' in the same information set. That is, a strategy must give the same distribution over actions to all histories in the same information set. If player i is following σ_i, then whenever a history h is reached where $P(h) = i$, player i samples from $\sigma_i(h)$ to choose its action. We let Σ_i be the set of possible strategies for player i. A **strategy profile**, σ, consists of a strategy for each player, $\sigma_1, \ldots, \sigma_n$, with σ_{-i} referring to all the strategies in σ except σ_i. We define $u_i(\sigma)$ to be the expected reward for player i when all players play according to σ. Abusing notion, we let $u_i(\sigma_1, \sigma_2) = u_i(\{\sigma_1, \sigma_2\})$ and $u_i(\sigma_{-i}, \sigma'_i) = u_i(\sigma_{-i} \cup \{\sigma'_i\})$.

2.2 Best Response and Nash Equilibria

If player i knows the strategies of the other players, then it can compute a utility maximizing response. Given σ_{-i}, we say player i's **best response** is any strategy that maximizes

$$b_i(\sigma_{-i}) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma_{-i}, \sigma'_i), \tag{1}$$

where $b_i(\sigma_{-i})$ is called the best response value to σ_{-i}. If the other player's strategies are not known, one usually appeals to the Nash equilibrium solution concept.

DEFINITION 2 (NASH EQUILIBRIUM). *A* ***Nash equilibrium*** *is a strategy profile* σ *where for all players*

$$u_i(\sigma) = b_i(\sigma_{-i}) \tag{2}$$

An approximation of a Nash equilibrium or ***ε-Nash equilibrium*** *is a strategy profile* σ *where for all players*

$$u_i(\sigma) + \varepsilon \geq b_i(\sigma_{-i}) \tag{3}$$

If σ is a Nash equilibrium, then every player is playing a best response to σ_{-i} and therefore no player can benefit by deviating its strategy from σ. At an ε-equilibrium, no player can benefit more than ε by deviating from σ. All finite games have at least one Nash equilibrium. In the case of zero-sum extensive games with perfect recall there are efficient techniques for computing an ε-Nash equilibrium, such as linear programming [8], the excessive gap technique [3], and counterfactual regret minimization [14]. To clarify, when we talk of an equilibrium, we are referring to the strategy profile. When we talk of an equilibrium strategy, we are referring to a strategy belonging to an equilibrium.

An equilibrium strategy in a zero-sum game maximizes the agent's worst case utility over the possible strategies of its opponent. This can also be thought of as minimizing the best case loss of the agent. Mathematically, this equivalence is stated in the Minimax Theorem [9].

THEOREM 1 (MINIMAX THEOREM). *For any zero-sum extensive game* Γ*, we have*

$$v^* = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) = \min_{\sigma_2 \in \Sigma_2} \max_{\sigma_1 \in \Sigma_1} u_1(\sigma_1, \sigma_2) \tag{4}$$

where v^* *is the value of* Γ *for player* 1*.*

As a corollary we have for all equilibrium σ and all $\sigma'_1 \in \Sigma_1$, $\sigma'_2 \in \Sigma_2$

$$v^* = b_1(\sigma_2) \leq u_1(\sigma_1, \sigma'_2), \text{ and} \tag{5}$$

$$-v^* = b_2(\sigma_1) \leq u_2(\sigma_2, \sigma'_1) \tag{6}$$

An equilibrium strategy is guaranteed to obtain at least the zero-sum game's value for player i. An approximate equilibrium strategy can be evaluated by how far from this value an opponent can shift the game. We call this difference a strategy's exploitability.

DEFINITION 3. *We define the* ***exploitability of player i*** *for a strategy* σ_i*, written* $\varepsilon_i(\sigma_i)$*, as*

$$\varepsilon_1(\sigma_1) = b_2(\sigma_1) + v^*, \text{ and} \tag{7}$$

$$\varepsilon_2(\sigma_2) = b_1(\sigma_2) - v^* \tag{8}$$

Exploitability measures how much an opponent who knows σ_i can benefit by player i's failure to play an equilibrium strategy. It is a worst case bound on the quality of a strategy. This makes exploitability the natural metric for evaluating any technique whose goal is to compute a strong strategy with no knowledge of how an opponent might play. Note that for all σ_i, $\varepsilon_i(\sigma_i) \geq 0$ and for all equilibria σ^*, $\varepsilon_i(\sigma_i^*) = 0$.

2.3 Game Abstraction

The goal of this paper is to evaluate the use of abstraction to turn intractably large extensive games into smaller, tractably sized games. In particular, how does abstraction affect the quality of the resulting strategies in the full game? There are two obvious methods for creating a smaller game from a larger one. First, we can merge information sets together. Second, we can restrict the actions a player can take from a history. We can also do a combination of these two methods.

DEFINITION 4 (ABSTRACTION). *An **abstraction for player i** is a pair $\alpha_i = \langle \alpha_i^{\mathcal{I}}, \alpha_i^A \rangle$, where,*

- *$\alpha_i^{\mathcal{I}}$ is a partitioning of H_i, defining a set of abstract information sets that must be coarser[1] than $\mathcal{I}_i$, and*
- *α_i^A is a function on histories where $\alpha_i^A(h) \subseteq A(h)$ and $\alpha_i^A(h) = \alpha_i^A(h')$ for all histories h and h' in the same abstract information set. We will call this the abstract action set.*

*The **null abstraction** for player i, is $\phi_i = \langle \mathcal{I}_i, A \rangle$. An **abstraction** α is a set of abstractions α_i, one for each player. Finally, for any abstraction α, the **abstract game**, Γ^α, is the extensive game obtained from Γ by replacing $\mathcal{I}_i$ with $\alpha_i^{\mathcal{I}}$ and $A(h)$ with $\alpha_i^A(h)$ when $P(h) = i$, for all i.*[2]

Strategies for abstract games are defined in the same manner as for unabstracted games, but the function must assign the same distribution to all histories in the abstraction information set, and assign zero probability to actions not in the abstract action set. We will use the notation Σ_i^α to refer to the set of strategies for player i in the abstract game Γ^α.

3. MONOTONICITY

As we have discussed, the traditional approach for finding good strategies in very large zero-sum extensive games is to use abstraction. Given limited resources, an abstraction is identified (by hand or automatically) such that solving the resulting abstract game is tractable. As more resources become available or solvers become more efficient, the abstraction can be further refined, resulting in larger abstract games. The presumption of this approach is that since the refined game is more representative of the full game, a Nash equilibrium in this refined game would be closer to equilibrium in the full game. In other words, refining an abstraction leads to monotonic improvement in strategy quality.

[1]Recall that partition A is coarser than partition B, if and only if every set in B is a subset of some set in A, or equivalently x and y are in the same set in A if x and y are in the same set in B.

[2]As discussed earlier some partitions of information can result in situations where a player is forced to forget their own past decisions. This paper only considers abstractions that result in an abstract game with perfect recall.

The dramatic advancements that have come out of the AAAI Poker Competitions, suggests this may be true. In the first year of the event, teams used linear programming techniques to solve small games, containing only a subset of the betting rounds [1, 4]. These small abstract games were then pieced together to create an overall poker strategy. In the following years, new equilibrium solving techniques were introduced and teams were able to solve a single abstract game to find a complete strategy [3, 13, 14]. In the following year, further enhancements were made to solve even larger abstract games, with as many as 10^{12} game states. Each year, (roughly speaking) solving the largest abstract game has won the competition. Hence, a strategy of merely building and solving the largest possible (carefully constructed) abstracted games has been employed by the top teams, with the presumption that this will monotonically lead to stronger poker strategies. In fact, the success of this approach in the competition is evidence that the basic presumption may be true.

In this section, we formalize this monotonicity assumption of abstraction. We present a definition for refining abstractions and then state a number of possible strong and weak monotonicity properties that may hold. Unfortunately, every useful form of this property does not hold in general. In the next section, we give counterexamples, showing even in a small poker game that a plethora of non-monotonicities in abstractions can exist.

We begin by defining a notion of abstraction refinement.

DEFINITION 5 (REFINEMENT). *Let α_i and β_i be abstractions for player i. We say α_i **refines** β_i, written $\alpha_i \sqsupseteq \beta_i$ if and only if $\beta_i^{\mathcal{I}}$ is coarser than $\alpha_i^{\mathcal{I}}$ and $\alpha_i^A(h) \supseteq \beta_i^A(h)$ for all $h \in H_i$. If $\alpha_i \sqsupseteq \beta_i$ and $\beta_i \sqsupseteq \alpha_i$ then we say α_i and β_i are **equivalent**, written $\alpha_i \equiv \beta_i$. In addition, we say α refines β, or $\alpha \sqsupseteq \beta$, if and only if $\alpha_i \sqsupseteq \beta_i$ for all i.*

A refinement, therefore, allows the player's decisions to be contingent on all the same information (and possibly more). A refinement also allows a player to make all of the same choices (and possibly more). As a result, a strict refinement necessarily gives the player more available strategies.

THEOREM 2. *If $\alpha \sqsupseteq \beta$ then for all i, $\Sigma_i^\alpha \supseteq \Sigma_i^\beta$.*

PROOF. Suppose $\sigma_i \in \Sigma_i^\beta$. For any h and h' in the same abstract information set of α_i, they must also be in the same abstract information set of β_i since $\beta_i^{\mathcal{I}}$ is coarser than $\alpha_i^{\mathcal{I}}$. Therefore $\sigma_i(h) = \sigma_i(h')$. Furthermore, since $\sigma_i(h)$ defines a distribution over $\beta_i^A(h)$ it also defines a distribution over the superset $\alpha_i^A(h)$. Thus $\sigma_i \in \Sigma_i^\alpha$. ∎

One might presume that more available strategies would mean better strategies would be found. However, solving games does not mean finding the best strategy from the available set, but rather finding a pair of strategies, one for each player, that are in equilibrium. The abstraction for each player matters. Again, one might presume that giving more available strategies to all players would result in a strategy that is closer to equilibrium in the full game. We can state these presumptions formally, using our notion of exploitability, $\varepsilon_i(\sigma_i)$, as the quantitative value for how far σ_i is from an equilibrium strategy.

PROPERTY 1 (MONOTONICITY). *Let α and β be abstractions, where $\alpha \sqsupseteq \beta$.*

(a) ***Strong Monotonicity for player i:*** *If σ^α and σ^β are Nash equilibria for Γ^α and Γ^β (respectively), $\varepsilon_i(\sigma_i^\alpha) \leq \varepsilon_i(\sigma_i^\beta)$.*

(b) ***Weak Monotonicity for player i:*** *Let $\Sigma^{\alpha,*}$ and $\Sigma^{\beta,*}$ be the set of all Nash equilibria for Γ^α and Γ^β (respectively), $\min_{\sigma \in \Sigma^{\alpha,*}} \varepsilon_i(\sigma_i) \leq \min_{\sigma \in \Sigma^{\beta,*}} \varepsilon_i(\sigma_i)$.*

*If the (strong or weak) monotonicity property for player i holds only if $\alpha_{-i} \equiv \beta_{-i}$ (*i.e.*, we only refine the player's abstraction), then we call it **player monotonic**. If it only holds if $\alpha_i \equiv \beta_i$ (*i.e.*, we only refine the opponent's abstraction) then we call it **opponent monotonic**.*

Strong monotonicity requires that every equilibrium strategy in the finer abstract game be less exploitable than every equilibrium strategy in the coarser abstract game. This is the most hopeful property, and the one that is presumed by work that seeks to use larger and larger abstractions (since most game solvers make no claims about the equilibria they find or approximate). Extensive games, though, may have many equilibria, and some of these may be closer to optimal in the full game than others. Weak monotonicity states that the best equilibrium strategy in the finer abstract game is less exploitable than the best equilibrium strategy in the coarser abstract game. Player and opponent (weak and strong) monotonicity are weaker monotonicity properties that refer to when a refinement only affects the player's or opponent's abstraction, respectively.

Unfortunately, none of these monotonicity properties hold in general. We show specific counterexamples of all of them in the next two sections. However, before moving on to these counterexamples, there is one condition under which we can prove monotonicity does hold. If we use an abstraction where the opponent plays in the null abstraction (*i.e.*, the full game), then we see monotonicity in the player's abstraction.

THEOREM 3. *Let α and β be abstractions where $\alpha \sqsupseteq \beta$ and $\alpha_2 \equiv \beta_2 \equiv \phi_2$. If σ^α and σ^β are Nash equilibria for Γ^α and Γ^β, respectively, then $\varepsilon_1(\sigma_1^\alpha) \leq \varepsilon_1(\sigma_1^\beta)$.*

PROOF. We use the definition of ε_1 (steps 9 and 12) and the Minimax Theorem (steps 10 and 12), plus the fact that a maximum over a set is no larger than the maximum over a superset (step 11) to prove this theorem.

$$v^* - \varepsilon_1(\sigma_1^\alpha) = \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1^\alpha, \sigma_2) \tag{9}$$

$$= \max_{\sigma_1 \in \Sigma_1^\alpha} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) \tag{10}$$

$$\geq \max_{\sigma_1 \in \Sigma_1^\beta} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) \tag{11}$$

$$= \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1^\beta, \sigma_2) = v^* - \varepsilon_1(\sigma_1^\beta). \tag{12}$$

Removing v^* followed by negating the sides of the inequality and flipping the $\geq$ to $\leq$ gives us our result. ∎

Unfortunately, unless there is considerable asymmetry in the number of choices available to the two players, solving a game where even one player operates in the null abstraction is typically infeasible. This is certainly true in the large poker games that have been examined recently in the literature. Although the result gives some insight as to when abstraction is safe, it does not give a foundation for the practical use of abstraction.

4. COUNTEREXAMPLES

In this section we present counterexamples to the monotonicity property. We performed several tests on different types of abstractions in a small poker variant. In these tests, we focus on the first player and look at abstracting both its view and its opponent's view. After abstraction, we solve the resulting abstract game and then calculate the best response value to the player's strategy in the full game. Finally, the value of the full game (which is non-zero due to a positional advantage for player two) is removed from this value to obtain the exploitability of the player's strategy. To reiterate, these experiments can only be performed in a game small enough that we can solve it exactly.

4.1 Leduc Hold'em

Texas Hold'em games, which are the focus of much of the recent research, are too large to feasibly compute the exploitability of a strategy. As a consequence, we chose to use a smaller poker game called Leduc Hold'em [11] for our experiments. In this game the deck contains two Kings, two Queens and two Jacks. Each player initially pays one chip to the pot, called an ante, and is dealt a single private card. After a round of betting, a community card is dealt face up. After a subsequent round of betting, if neither player has folded, both players reveal their private cards. If either player pairs their card with the community card they win the chips in the pot. Otherwise, the player with the highest private card is the winner. In the event both players have the same private card, they draw and split the chips in the pot.

Each betting round starts with player 1's action. Player 1 may *check* or *bet*. Actions then alternate between the two players. When facing a bet, a player can *fold*, *call* or *raise*. Folding forfeits the chips in the pot to the opponent and the hand ends. Calling requires the player to match the bet faced and the betting round ends. Raising requires a player to put more chips into the pot than the current bet faced. The opponent must then respond with an action of its own. If player 1's first action is a check, then player 2 has the option of checking to end the betting round.

In limit Leduc Hold'em the total number of bets and raises in a round is not allowed to exceed two. Furthermore, each bet in the first round is fixed to be two chips. The subsequent betting round has a fixed size of four chips. In no-limit Leduc Hold'em a player may bet or raise any amount of his remaining chips. A raise is only valid if it is at least the size of the preceding bet or raise on the same round. There is no limit on the number of times a player may bet or raise other than the number of chips remaining in their stack.

For our experiments, we report the exploitability of a strategy in millibets per hand (mb/h). This reflects how much a player is expected to lose per hand in the worst case. A **millibet** is one thousandth of a small bet, which in limit is two chips and in no-limit is one chip. Equilibrium strategies were computed with CPLEX using a sequence form representation [8]. Though there exists many equilibrium strategies, we only computed a single strategy for each abstract game. Since Leduc Hold'em is such a small game, we can compute exploitability exactly. That is, the exploitability numbers for the computed strategies have no variance as sampling was not required. Some error is derived from the actual equilibrium calculations, which is on the order of 10^{-5} mb/h or smaller.

Abstraction	Exploitability
FULL-FULL	0.0
J.Q.K-FULL	55.2
JQ.K-FULL	69.0
J.QK-FULL	126.3
JQK-FULL	219.3
JQ.K-JQ.K	272.2
JQ.K-J.Q.K	274.1
FULL-J.QK	345.7
FULL-JQ.K	348.9
J.Q.K-J.Q.K	359.9
J.Q.K-JQ.K	401.3
J.QK-J.QK	440.6
FULL-JQK	459.5
FULL-J.Q.K	491.0
JQK-JQK	755.8

Table 1: Exploitability of various limit Leduc Hold'em strategies in millibets/hand (mb/h)

4.2 Card Abstraction

The first example we examine involves abstracting the cards in limit Leduc Hold'em. Card abstraction is typically done by grouping together hands of similar *strength* or distribution over future strengths. In the abstract game an agent cannot distinguish between the hands in a group. The intuition is that hands of similar current or future strength can be played with a similar strategy. Grouping hands in this fashion requires a strength metric, typically placing some weight on the expected probability of winning at a showdown as well as on the variance of this probability. When using this method to abstract cards, the size of the abstract game is determined by the size of each hand group.

Johanson [7] describes a method for creating abstractions using a metric called *hand strength squared*, which creates roughly equal sized groups based on a metric that incorporates both the expectation and variance of future strength. A method by Gilpin *et al.* [5] uses a bottom-up approach to create a potential-aware abstraction. This method works by clustering the terminal nodes and then repeatedly clustering earlier rounds based upon *distances* between distributions over future clusters. Both of these methods have proven to be effective in poker competitions.

Limit Leduc Hold'em.

For our experiments in Leduc Hold'em, we start by abstracting the initial deal of the private card. For example, normally the player can separate their private card into King (K), Queen (Q), or Jack (J). One abstracted view is that the private card is either a King (K) or not a King (JQ). Similarly, an abstracted view of the community card is either the private card was paired (*pair*), or it was not (*nopair*). The abstractions we used in our experiment are *FULL*, *J.Q.K/-pair.nopair*, *JQ.K/pair.nopair*, *J.QK/pair.nopair*, and *JQK/-pair.nopair*. Here, *FULL* denotes the null abstraction, while periods between the cards separate information sets. For simplicity's sake, we drop *pair.nopair* from the description of the abstractions as it does not add ambiguity. Note that *J.Q.K* refines both *JQ.K* and *J.QK*, both of which refine *JQK*. A subset of the results of this experiment are displayed in Table 1. The abstraction name in this table describes the abstraction used by the player followed by a "-" and the abstraction used by the opponent.

As is required by Theorem 3, monotonicity holds when our opponent uses the *FULL* abstraction. However, there are several examples where monotonicity fails to hold when our opponent is abstracted. For instance, *JQ.K-JQ.K* is only exploitable by 272.2 mb/h whereas *JQ.K-J.Q.K* is exploitable by 274.1 mb/h. In turn, this strategy is even less exploitable than the one for *J.Q.K-J.Q.K*, which is exploitable by 359.9 mb/h. These are explicit counterexamples to both strong player monotonicity and strong opponent monotonicity. In other words, refining the card abstractions for both players, as is classically done for competitions, can produce a more exploitable strategy.

4.3 Betting Abstraction

The second example we examine involves abstracting betting options in no-limit Leduc Hold'em. Betting abstraction is typically done by restricting the number of betting options from each history. Each additional option has an exponential effect on the size of the game, so often only two or three options are available. A pot sized bet and an all-in bet are frequently among the ones chosen. When using the abstract strategy to play in the full game, an agent must *translate* the size of the bets in the game to and from actions in the abstract game. For example, if an opponent bets five chips when four chips were in the pot, the agent would likely translate this bet to a pot bet, or four chips. This translation can be particularly troublesome as an opponent can make awkward sized bets that do not map well to the abstract game. These bets distort the agent's belief of the number of chips in the pot and can lead to poor decision making.

Gilpin and colleagues [6] describe the action abstraction and translation for their no-limit agent used in the 2007 AAAI Computer Poker Competition. The main concept is that if our opponent makes a bet of size b that we do not understand, we compare that size to the surrounding bet sizes we do understand, for instance a and c, and translate it to the *closest* one. If $a < b < c$ then we define the closer size to be whichever ratio of $\frac{b}{a}$, $\frac{c}{b}$ is smallest. This allows us to play an abstract strategy in the full game.

Unlike a true no-limit game where each betting option is some number of chips, the betting options in our game are measured in terms of pot-fractions. Thus, a 50% pot bet would allow us to bet 50% of the current pot size. The exception to this is the all-in option, which allows a player to bet all of their remaining chips.

No-Limit Leduc Hold'em.

In the game we created, each player starts with 12 chips and is allowed 12 different pot-fraction bets which range from 25% pot to all-in. In the abstract games, each player has up to four betting options. These options are 50% pot (h), 100% pot (p), 200% pot (d) and all-in (a). Different abstractions are created by disallowing one or more of h, p, and d. When h or d are disallowed, any bets that would have been translated to them are instead translated to a p bet. Similarly, if p is also disallowed then all bets are translated to a. This ensures that an abstraction with more bets, like *hpda*, is a proper refinement of an abstraction with fewer bets, like *pda*. Note that this is not the translation described by Gilpin *et al.* as their technique may not refine an abstraction with the addition of new betting choices. The betting abstractions we use are *FULL*, *hpda*, *hpa*, *pda*, *pa*, and *a*. A subset of the results are displayed in Table 2.

Again, we see that when our opponent uses the *FULL*

Abstraction	Exploitability
FULL-FULL	0.0
hpda-FULL	0.4
pda-FULL	0.4
hpa-FULL	2.8
pa-FULL	2.8
a-FULL	5.7
hpda-hpda	94.0
FULL-hpda	103.2
a-hpda	116.8
pda-hpda	118.5
hpa-hpda	135.2
pa-hpda	138.3
hpa-hpa	185.4
pa-hpa	185.8
pda-hpa	190.4
FULL-hpa	193.7
a-a	238.0
pa-a	246.2
FULL-a	247.7
pda-pda	255.4
FULL-pda	276.1
pa-pa	373.3
FULL-pa	390.4

Table 2: Exploitability of various no-limit Leduc Hold'em strategies in millibets/hand (mb/h)

abstraction that the exploitability of our various abstractions are ordered correctly. We also see that this ordering breaks as soon as our opponent is abstracted. One example is that *a-a* is only 238.0 mb/h exploitable while *pa-a* is 246.2 mb/h exploitable. Similarly, *pa-pa* is even more exploitable at 373.3 mb/h. Again, these are counterexamples to strong player and strong opponent monotonicity. In other words, giving the player or the opponent more actions can result in a more exploitable strategy.

5. QUALITY OF ABSTRACT EQUILIBRIA

As alluded to earlier, not all abstract equilibrium strategies are equivalent in terms of exploitability in the full game. Strong monotonicity requires all equilibria in a finer abstraction to be stronger than all equilibria in a coarser abstraction. However, maybe some equilibria are weaker, while some equilibria are stronger. This is the premise of our weak monotonicity property. In this section we look into the best equilibrium for a particular abstraction, which first requires some new algorithmic development.

We begin by introducing **sequence form**, which is a compact representation of a zero-sum extensive game. A game in sequence form is described by the following components:

- An m by n **payoff matrix** A. We call the row player x and the column player y. Entry (i, j) of A is the reward for player x if x plays sequence form action i and y plays sequence form action j.
- A constraint matrix E and vector e. We say x is a valid sequence form strategy if $Ex = e$ and $x \geq 0$.
- A constraint matrix F and vector f. This pair is the analog of E and e for player y.

Given a sequence form game and a strategy x, one can compute $b_y(x)$, y's best response value to x, with the following linear program:

$$b_y(x) = \min_u u^T f \quad \textbf{subject to} \quad u^T F \geq -x^T A \tag{13}$$

Following the best response linear program, we can derive a linear program for computing a Nash equilibrium strategy for x as:

$$b_y(x^*) = \min_{u,x} u^T f \quad \textbf{subject to} \quad u^T F \geq -x^T A \quad Ex = e \quad x \geq 0 \tag{14}$$

For more information about sequence form refer to *Efficient Computation of Behavior Strategies* [12].

We can efficiently find the best abstract equilibrium strategy using the following linear program:

$$\min_{u,v,x} v^T g \quad \textbf{subject to} \quad v^T G \geq -x^T B \quad u^T F \geq -x^T A \quad u^T f = t^* \quad Ex = e \quad x \geq 0 \tag{15}$$

Here (A, E, e, F, f) is the sequence form description of Γ^α and (B, E, e, G, g) is the sequence form description of a new game Γ^β. In Γ^β, $\beta_1 \equiv \alpha_1$ and $\beta_2 \equiv \phi$. t^* is the value of Γ^α.

THEOREM 4. *Any x^* that is part of an optimal solution to (15) is an equilibrium strategy in Γ^α and no other equilibrium strategy of Γ^α is less exploitable in Γ.*

PROOF. Let (u^*, v^*, x^*) be an optimal solution to (15). Clearly, (u^*, x^*) is a feasible point of (14) as the constraints of (14) are a subset of the constraints of (15). Furthermore, this point is optimal as (14) has optimal value t^* and the objective function, $u^T g$, is constrained in (15) to meet this condition. This implies x^* is an equilibrium strategy of Γ^α. Let x_0 be any equilibrium strategy of Γ^α. We can find u_0 and v_0 using (13) in Γ^α and Γ respectively. This implies (u_0, v_0, x_0) is a feasible point of (15). We note that if we only let v vary in (15) it is equivalent to (13) and therefore the value at (u_0, v_0, x_0) is $b_y(x_0)$. Finally, the value of (15) at a feasible point must be no less than the value at an optimal point. Therefore, $b_y(x^*) \leq b_y(x_0)$, which implies x_0 can be no less exploitable than x^*. We note that the best response values in this proof are in Γ^β, but since our opponent's abstraction is ϕ, these best response values are the same as in Γ. ∎

Unfortunately solving (15) requires as much work as solving Γ^β, which involves a full representation of one player's strategy space. However, we can still use this approach to examine abstractions in small games, allowing us to explore the weak monotonicity property. Solving for the worst case equilibrium strategy, although interesting, does not appear to be tractable for even small games. Therefore, we do not examine it in this work.

5.1 Best Equilibria using Card Abstraction

Though we cannot tractably compute the best equilibrium strategies in many games we are interested in, we can examine them in Leduc Hold'em. In Table 3 we show a subset of the results of an experiment, which uses the same card abstractions from our first limit experiment. A new column is added to Table 1, which refers to the exploitability of the *best* equilibrium strategy.

We find that even when we use the *best* equilibrium that we still see counterexamples to monotonicity. The exploitability of the *best* equilibrium strategy in the *J.Q.K-J.Q.K* abstract game is 358.6 mb/h, whereas the *best* equilibrium for *JQ.K-J.Q.K* is only exploitable by 78.8 mb/h, violating

Abstraction	Exploitability	Exploitability (Best)
FULL-FULL	0.0	0.0
FULL-J.Q.K	491.0	1.7
FULL-JQK	459.5	10.1
FULL-J.QK	345.7	45.3
J.Q.K-FULL	55.2	55.2
FULL-JQ.K	348.9	57.7
JQ.K-FULL	69.0	69.0
JQ.K-J.Q.K	274.1	78.8
J.Q.K-JQ.K	401.3	88.8
J.QK-FULL	126.3	126.3
JQK-FULL	219.3	219.3
JQ.K-JQ.K	272.2	272.2
J.Q.K-J.Q.K	359.9	358.6
J.QK-J.QK	440.6	440.6
JQK-JQK	755.8	710.2

Table 3: Exploitability of various limit Leduc Hold'em strategies in millibets/hand (mb/h)

weak player monotonicity. Perhaps even more disturbing is the fact that *JQ.K-JQ.K*, which is obtained from the previously mentioned game by merging the Jack and Queen for both players, is only exploitable by 272.2 mb/h. In this case, even if we managed to converge to the best strategy in the larger game it is still worse than one we happened to converge to in the smaller game.

When ordered by the least exploitable equilibrium strategy we see many opposite trends as when ordered by the exploitability of the strategy that the LP solver happened upon. First, if we could indeed find the best strategy, we would like to give our player as much information about the game as possible. There exists an equilibrium strategy in the abstract game *FULL-J.Q.K* that is only exploitable by 1.7mb/h, however our equilibrium solver happened upon one exploitable by 491mb/h. This difference is incredible when you consider we can guarantee ourselves a strategy that is exploitable by 219.3mb/h when using *JQK-FULL*, in which our player does not have any knowledge of its own private card prior to the community card's arrival! Second, even when giving our player as many options as possible, there is a counterexample to weak opponent monotonicity as we would rather give our opponent *JQK* than the refined *J.QK*. Finally, there is an example of both giving ourselves or our opponent more options leading to higher best case exploitability. Specifically, *JQ.K-J.Q.K* is better than *J.Q.K-J.Q.K* and *J.Q.K-JQ.K* is better than *J.Q.K-J.Q.K*.

6. DISCUSSION

We used a smaller game, Leduc Hold'em, in this paper so that we could analyze abstract game strategies in terms of their exploitability in the full game. However, we did not carefully construct this game or the abstractions to create the counterexamples we have presented. Leduc Hold'em is scaled version of Texas Hold'em in which the number of cards was reduced and the betting is simplified. The counterexamples arose naturally from the same kind of abstractions used in Texas Hold'em – card abstractions that group hands with similar *strength* and *potential* and restricting the betting actions. Since these pathologies are common in this small poker game, we expect them to also be common in larger poker games.

We now introduce an even simpler game to further illustrate how these pathologies arise. Consider the two-player,

	a	b	c	d
A	7	2	8	0
B	7	10	5	6

Figure 1: An example two-player, zero-sum matrix game. The entries are the row player's utilities.

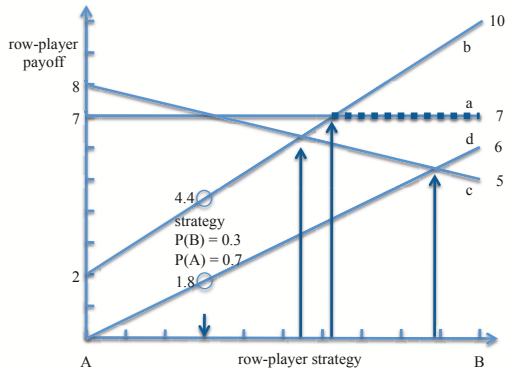

Figure 2: Visual representation of the matrix game in Figure 1.

zero-sum matrix game shown in Figure 1. Matrix games are a special case of extensive games where players simultaneously choose an action. Their joint decision identifies an entry in the matrix specifying the utility for the *row player*, where the utility for *column player* is simply the negation. In this game, the row player has two actions, A and B, and the column player has four actions, a, b, c, and d. An abstraction of this game is obtained by eliminating one or more actions for either player.

Figure 2 gives a visual representation of this game. The x-axis shows the row player's strategy space. For example, a point 30% along the x-axis represents a strategy where the row player selects action B with probability 0.3 and action A the remainder of the time. Each line corresponds to a different action for the column player (a, b, c or d). The height of the line above any point on the x-axis represents the row player's utility when the row player plays that particular strategy and the column player plays the action represented by the line. For example, if the row player plays the strategy $P(B) = 0.3$ and the column player selects action b, then the utility for the row player is approximately 4.4, whereas if the column-player selects action d the utility for the row player is approximately 1.8. These points are circled in Figure 2. An equilibrium strategy guarantees the best worst-case value, which in the full game occurs at the intersection of lines c and d, giving at least a value of slightly more than 5 to the row player.

Consider an abstract game where the column player only has actions a and b. This abstract game has a range of equilibrium strategies for the row player corresponding to the dash-emphasized segment of line a. Not only are there many equilibrium strategies in this abstract game, but some of them are *better* than others in terms of their worst-case utility in the full game. Since the true equilibrium strategy for the row player lies within this space, this abstract game has a solution that is not exploitable. Now consider a refinement of this abstraction where the column player has action c in addition to a and b. Here, the equilibrium strategy for

the row player shifts to the left to the intersection of lines b and c. The addition of this action removes the true equilibrium strategy from the set of equilibrium strategies in the abstract game. More importantly, this unique equilibrium strategy has a lower worst-case utility in the full game than any equilibrium strategy in the coarser abstract game. Why does this occur? In essence, providing an additional action to an opponent has made the stronger strategies (including the true equilibrium) appear less attractive as the opponent lacks the ability play strategies that would show that the alternatives are indeed worse.

Now, we show a pathology that arises from abstracting the row player's actions. Suppose that the row player is only allowed to take action B and the column player can only take action c. The equilibrium strategy is the only strategy for the row player, $P(B) = 1$, and the row player has a worst-case utility of 5. If we refine the game to allow the row player to also take action A, the equilibrium strategy moves to $P(A) = 1$, so the row player increases its utility to 8. However, in the full game this strategy has utility 0 for the row player in the worst-case. Why does this occur? Providing additional strategies to a player can encourage the player to exploit the limitations of the opponent's abstraction, resulting in a strategy that is more exploitable by actions that become available to the opponent in the full game.

7. CONCLUSION

We have shown that the quality of strategies found in the refining of abstract games is dubious at best. Though the trends in the AAAI Computer Poker Competition seem to suggest that monotonic properties hold when using abstraction techniques, we have demonstrated that they do not hold in general and appear very unlikely to hold in the Texas Hold'em games used in the competition. The effects of the typical abstraction techniques on the strategies produced is not predictable. Furthermore, creating larger abstract games is not guaranteed to improve the quality of the strategies found. Also, different solutions to the same abstract game are not equally strong in the full game. We can no longer solely blame an abstraction technique if a poor strategy is produced.

With this result, we are now aware of the possible consequences when using abstraction in extensive games. Though the consequences can be dire, we are still forced to abstract when solving large games for the foreseeable future. As such, new techniques and methods of evaluating abstractions are required. Most importantly, more careful selection of equilibrium in abstract games could dramatically impact the strength of strategies that are computed.

Acknowledgments

We would like to thank the Computer Poker Research Group at the University of Alberta for helpful discussions related to this work.

8. REFERENCES

[1] 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 *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[2] S. Ganzfried and T. Sandholm. Computing an approximate jam/fold equilibrium for 3-agent no-limit texas hold'em tournaments. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, 2008.

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

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

[5] A. Gilpin and T. Sandholm. 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, 2007.

[6] A. Gilpin, T. Sandholm, and T. B. Sorensen. 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*, 2008.

[7] M. Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, 2007.

[8] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94:167–215, 1997.

[9] J. V. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

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

[11] 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. AUAI Press, 2005.

[12] B. V. Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14:220–246, 1996.

[13] M. Zinkevich, M. Bowling, and N. Burch. A new algorithm for generating equilibria in massive zero-sum games. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI)*, pages 788–793, 2007.

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

[15] M. Zinkevich and M. Littman. The AAAI computer poker competition. *Journal of the International Computer Games Association*, 29, 2006. News item.