Extensive-Form Game Abstraction With Bounds

CHRISTIAN KROER, Carnegie Mellon University
TUOMAS SANDHOLM, Carnegie Mellon University

Abstraction has emerged as a key component in solving extensive-form games of incomplete information. However, lossless abstractions are typically too large to solve, so lossy abstraction is needed. All prior lossy abstraction algorithms for extensive-form games either 1) had no bounds on solution quality or 2) depended on specific equilibrium computation approaches, limited forms of abstraction, and only decreased the number of information sets rather than nodes in the game tree. We introduce a theoretical framework that can be used to give bounds on solution quality for any perfect-recall extensive-form game. The framework uses a new notion for mapping abstract strategies to the original game, and it leverages a new equilibrium refinement for analysis. Using this framework, we develop the first general lossy extensive-form game abstraction method with bounds. Experiments show that it finds a lossless abstraction when one is available and lossy abstractions when smaller abstractions are desired. While our framework can be used for lossy abstraction, it is also a powerful tool for lossless abstraction if we set the bound to zero.

Prior abstraction algorithms typically operate level by level in the game tree. We introduce the extensive-form game tree isomorphism and action subset selection problems, both important problems for computing abstractions on a level-by-level basis. We show that the former is graph isomorphism complete, and the latter NP-complete. We also prove that level-by-level abstraction can be too myopic and thus fail to find even obvious lossless abstractions.

Categories and Subject Descriptors: I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; J.4.a [**Social and Behavioral Sciences**]: Economics

General Terms: Algorithms, Economics, Theory

Additional Key Words and Phrases: Extensive-form game; abstraction; equilibrium finding

1. INTRODUCTION

Game-theoretic equilibrium concepts provide a sound definition of how rational agents should act in multiagent settings. To operationalize them, they have to be accompanied by techniques to compute equilibria, an important topic that has received significant attention in the literature [Littman and Stone 2003; Lipton et al. 2003; Gilpin and Sandholm 2007b; Jiang and Leyton-Brown 2011].

Typically, equilibrium-finding algorithms do not scale to very large games. This holds even for two-player zero-sum games (that can be solved in polynomial time [Koller et al. 1996]) when the games get large. Therefore, the following has emerged as the leading framework for solving large extensive-form games [Sandholm 2010]. First, the game is abstracted to generate a smaller game. Then the abstract game is solved for (near-)equilibrium. Then, the strategy from the abstract game is mapped back to the original game. Initially, game abstractions were created by hand, using domain dependent knowledge [Shi and Littman 2002; Billings et al. 2003]. More recently, automated abstraction has taken over [Gilpin and Sandholm 2006, 2007b; Zinkevich et al. 2007]. This has

This is an extended preprint version with appendices that show all proofs omitted from the published paper.

Authors addresses: C. Kroer and T. Sandholm, Computer Science Department, Carnegie Mellon University; email: {ckroer,sandholm}@cs.cmu.edu This material is based on work supported by the National Science Foundation under grants IIS-1320620, CCF-1101668, and IIS-0964579, as well as XSEDE computing resources provided by the Pittsburgh Supercomputing Center. The work was also supported by the Center for Computational Thinking at CMU, funded by Microsoft Research.

EC'14, June 8–12, 2014, Stanford, CA, USA. ACM 978-1-4503-2565-3/14/06 ...$15.00.

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

http://dx.doi.org/10.1145/2600057.2602905

typically been used for information abstraction, whereas action abstraction is still largely done by hand [Gilpin et al. 2008; Schnizlein et al. 2009]. Recently, automated action abstraction approaches have also started to emerge [Hawkin et al. 2011, 2012; Brown and Sandholm 2014].

Ideally, abstraction would be performed in a lossless way, such that implementing an equilibrium from the abstract game results in an equilibrium in the full game. Abstraction techniques for this were introduced by Gilpin and Sandholm [2007b] for a subclass of games called *game of ordered signals*. Unfortunately, lossless abstraction often leads to games that are still too large to solve. Thus, we must turn to lossy abstraction. However, significant abstraction *pathologies* (*nonmonotonicities*) have been shown in games that cannot exist in single-agent settings: if an abstraction is refined, the equilibrium strategy from that new abstraction can actually be worse in the original game than the equilibrium strategy from a coarser abstraction [Waugh et al. 2009a]! Until recently, all lossy abstraction techniques for general games of imperfect information were without any solution quality bounds. Basilico and Gatti [2011] give bounds for the special game class *Patrolling Security Games*. Johanson et al. [2013] provide computational methods for evaluating the quality of a given abstraction via computing a best response in the full game after the fact. Lanctot et al. [2012] present regret bounds for equilibria computed in imperfect recall abstractions, with their result also extending to perfect-recall abstractions. Their result depends on the counterfactual regret minimization (CFR) algorithm being used for equilibrium computation in the abstraction, and focuses on abstraction via information coarsening, thus allowing neither action abstraction nor reduction of the size of the game tree in terms of nodes. Sandholm and Singh [2012] provide lossy abstraction algorithms with bounds for stochastic games. They leave as an open problem whether similar bounds can be achieved for extensive-form games. A key complication keeping the analysis in their paper from directly extending to extensive-form games is information sets. In stochastic games, once the opponent strategies are fixed, the best response analysis can be approached on a node-by-node basis. With information sets this becomes much more complex, as strategies have to be evaluated not only according to how they perform at a given node, but also how they perform according to the distribution of nodes in the information set.

We develop analysis techniques that work on information sets as opposed to nodes, in order to answer this question in the affirmative. We develop the first bounds on lossy abstraction in general perfect-recall extensive-form games. These results are independent of the equilibrium computation approach, and allow both information coarsening and removal of nodes and actions to make the game tree smaller.

To achieve these more general results, we introduce several new analysis tools:

(1) Most importantly, much of our analysis operates on a per information set basis, which fundamentally changes how the analysis can be performed.
(2) To enable high expressiveness in the types of abstractions that can be computed, our mappings between real and abstract games are not all surjective functions.
(3) We introduce a stricter notion of strategy mapping, called *undivided lifted strategies*.
(4) We introduce a new solution concept, *self-trembling equilibrium*, the concept of which is used in our analysis to achieve bounds for general Nash equilibria.
(5) We introduce several new notions of error in abstractions that are used to characterize our bounds.

As a special case, our results generalize the results of Sandholm and Singh [2012], since any DAG-structured stochastic game can be converted to an extensive form game. We also tighten the bound given in their paper. Since our results apply to general Nash equilibria, as well as subgame-perfect equilibria, we also generalize the set of equilibrium concepts that the analysis applies to; theirs was for subgame-perfect equilibrium.

Our framework is the first general theoretical framework for reasoning about quality of equilibria computed in abstract games. The only prior work [Gilpin and Sandholm 2007b] is for a very restricted setting that closely resembles poker games, and only concerns lossless abstraction. In contrast, our results apply to all extensive-form games of perfect recall, and can be used to reason about both lossless and lossy abstraction. Further, we show that, even in the restricted setting considered in their paper, our framework can find lossless abstractions that theirs cannot.

We also present several new algorithmic characterizations and algorithms for building abstractions that minimize our bounds. We show several hardness results related to computing abstractions on a level-by-level basis, and show how only considering level-by-level abstraction can lead to not finding the smallest, or any, lossless abstractions. Motivated by these results, we develop the first abstraction algorithm with bounded loss, which operates on all levels of the tree at once. We then present experimental results, where we apply our algorithm to a relatively simple poker game. Finally, we discuss additional results, connections to other abstraction approaches, and directions for future research.

2. FRAMEWORK

Consider two extensive form games, the original game M and the abstract game M'. Both games have the same set of n players, denoted by N.

We let S denote the set of nodes in the game tree of M, and let $Z \subset S$ be the set of leaf nodes in M. $V_i(z)$ is the utility of player i for leaf node $z \in Z$. We let $\mathcal{I}$ denote the set of information sets, and denote a specific information set by I. We denote by A_I the set of actions at information set I. For any node $s \in I$, we also define $A_s = A_I$. We denote by σ_i a (behavioral) strategy for player i. For each information set I where it is the player's turn to move, it assigns a probability distribution over the actions in the information set, where $\sigma_i(I, a)$ is the probability of playing action a. We let $\sigma_i(s, a) = \sigma(I, a)$ for all $s \in I$. A strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$ has a behavioral strategy for each player. We will often use $\sigma(I, a)$ to mean $\sigma_i(I, a)$, since the information set uniquely specifies which player i is active.

As usual, randomness external to the players is captured by including an extra player called *nature* as a player. We denote it by 0, so $\sigma_0(s, a)$ is the probability that nature chooses outcome $a \in A_s$ at a given node s where nature is active. We use the subscript 0 to emphasize that this probability depends on nature only, not the strategy profile.

Let the probability of going from node s to node $\hat{s}$ under strategy profile σ be $\sigma[s \to \hat{s}] = \Pi_{\langle \bar{s}, \bar{a} \rangle \in \mathcal{P}_{s \to \hat{s}}} \sigma(\bar{s}, \bar{a})$ where $\mathcal{P}_{s \to \hat{s}}$ is the set of pairs of nodes and actions on the path from s to $\hat{s}$. We let the probability of reaching node s be $\sigma(s) = \sigma[r \to s]$, the probability of going from the root node r to s. Let $\sigma(I) = \sum_{s \in I} \sigma(s)$ be the probability of reaching any node in I.

Finally, for all behavioral strategies, the subscript $-i$ refers to the same definition, but without including player i.

The same elements are defined for M', except each one is denoted by a prime superscript. For example, the node space for M' is denoted by S'. The value for player i at leaf node $z' \in Z'$ in M' is denoted $W_i(z')$. We define the largest utility at any leaf node in the abstract game to be $\overline{W} = \max_i \max_{z \in Z} W_i(z)$, which will be useful later.

We assume that all payoffs are non-negative. This is without loss of generality since we can add a constant to all payoffs and keep the game strategically equivalent.

We let the height k of a node s be the length of the path from the node to the leaf level. Thus, the height of a leaf nodes is 0. We assume that all leaves are at the same depth in the tree, and at each height only one player has active information sets. This is without loss of generality since for any game that does not satisfy this, we can modify it by inserting singleton information sets with only one action available to stretch out the game until the property is satisfied; this keeps the game strategically equivalent. Let $\mathcal{H}_i$ be the set of heights where player i is active, and let $\mathcal{H}_i^l$ be $\{k : k \in \mathcal{H}_i, k \leq l\}$. Let S_k and $\mathcal{I}_k$ be the sets of nodes and information sets at height k, respectively.

For information set I and action $a \in A_I$ at level $k \in \mathcal{H}_i$, we let $\mathcal{D}_I^a$ be the set of information sets at the next level in $\mathcal{H}_i$ reachable from I when taking action a. Let t_a^s be the node transitioned to by performing action $a \in A_s$ at node s.

2.1. Abstraction functions

We define an information set abstraction function $f : \mathcal{I} \rightarrow \mathcal{I}'$ which maps from original information sets to abstract information sets. Similarly, $h : \mathcal{S} \rightarrow \mathcal{S}'$ is a node abstraction function that maps from original nodes to abstract nodes. Finally, $g : A \rightarrow A'$ is an action abstraction function that maps from original actions to abstract actions. All three functions are surjections, so that the game M' is no bigger than M, and ideally significantly smaller. We define $f^{-1} : \mathcal{I}' \rightarrow \mathcal{I}$ to be the inverse of the information set abstraction function. For a given I', $f^{-1}(I')$ returns all information sets that map to I'. We define h^{-1} and g^{-1} similarly. We also define the function $h_I^{-1} : \mathcal{S}' \rightarrow \mathcal{S}$ which for given $I, s' \in f(I)$ returns $h^{-1}(s') \cap I$, the subset of nodes in I that map to s'. Similarly, we define, $h_{\mathcal{P}_s}^{-1} : \mathcal{S}' \rightarrow \mathcal{S}$ which for a given s, s' returns $h^{-1}(s') \cap \mathcal{P}_s$, where $\mathcal{P}_s$ is the set of all ancestors and descendants of s.

We require that for any two actions that map to the same abstract, the two nodes respectively reached by performing the actions at a given node map to the same abstract node. Formally, for all $s \in S$, if $g(a_1) = g(a_2)$ for $a_1, a_2 \in A_s$, then $h(t_{a_1}^s) = h(t_{a_2}^s)$, where t_a^s is the node reached by performing action a at state s, as defined above.

As with prior abstraction approaches, these abstraction functions allow us to map a larger game onto a smaller game, through many-to-one mapping of nodes. We now present a novel approach for mapping nodes one-to-many. This leads to a larger set of valid abstractions, thereby sometimes allowing either smaller abstractions for a given bound, or the same size abstraction with a smaller bound. We call this "multi-mapping". An example where it is useful is given in Figure 1. In this game, nature starts out choosing among four actions. Player one then has two information sets, one for when either of the two leftmost nature actions were chosen, and one for when either of the two rightmost nature actions were chosen. If player 1 goes left at any node, player 2 then has two information sets. Player 2 can also only distinguish between nature's actions in sets of two, but different sets than player 1. In this game, we can losslessly treat player 2's information sets $\{2A, 2C\}$ and $\{2B, 2D\}$ as one big information set, since there is a mapping of actions at $2A$ to actions at $2B$ such that the utilities for Player 2 are the same, and similarly for $2C$ and $2D$. However, we need to formalize the notion that $2A$ is similar to $2B$ without merging the nodes, as they are not similar for Player 1. In mapping the smaller information set $\{2A, 2C\}$ onto the abstract information set $\{2A, 2C, 2B, 2D\}$, we wish to specify that $2A$ is represented by the two nodes $2A$ and $2B$. [1]

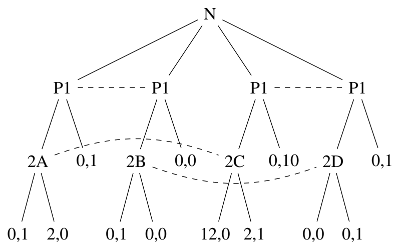


Fig. 1. Extensive form game, with nature playing first, picking among four outcomes uniformly at random. The next level, with nodes denoted P1, belongs to player 1, and the last level of internal nodes named 2A-2D belong to player 2.

We will now incorporate "multi-mapping" into the abstraction framework. To our knowledge, we are the first to allow for such abstraction. We define a mapping function $\phi : \mathcal{S}_\phi \rightarrow 2^{S'}$ which maps from a subset $\mathcal{S}_\phi \subset S$ of the original nodes to the powerset $2^{S'}$ of the abstract nodes. For all information sets I, we define the sets $I_\phi, I_{\neg\phi}$, the partition of I into two sets such that $s \in \mathcal{S}_\phi$ for $s \in I_\phi$ and $s \notin \mathcal{S}_\phi$ for $s \in I_{\neg\phi}$.

[1]This does not create a smaller abstraction in terms of the tree size, but it decreases the number of information sets. This is relevant to equilibrium-finding algorithms, such as CFR, whose complexity depends on the number of information sets rather than the number of nodes in the game tree. A similar notion of abstraction was introduced by Lanctot et al. [2012].

Since in our abstraction framework we are allowing "multi-mapping" (one original node mapped to multiple abstract nodes), we have to ensure that all those abstract nodes are similar (in terms of their dependence on the other players). Specifically, we require the following constraints for each information set I such that $I_\phi > 0$.

(1) If an abstract node $s' \in f(I)$ is part of $\phi(s)$ for some $s \in I_\phi$, then no other nodes in I can map to s'.
(2) For all nodes s and pairs of nodes $s'_1, s'_2 \in \phi(s)$ at level k, each level $l \in \mathcal{H}_{-0}, l > k$, the edges on the respective paths to s'_1, s'_2 must be in an information set together, and belong to the same action, or both be in information sets with a single action.
(3) For all abstract nodes $s' \in \phi(s)$, we require that the subtree rooted at s also maps onto the subtree rooted at s'. For all such pairs of nodes, we define a surjective function $\phi_{s,s'}$, completely analogous to the function h, defining this mapping.
(4) For all descendants $\hat{s}$ of s and pairs of nodes $s'_1, s'_2 \in \phi(s)$, $\phi_{s,s'_1}(\hat{s})$ must be in the same information set as $\phi_{s,s'_2}(\hat{s})$.

The example presented above can be described in terms of our ϕ function: $\phi(2A) = \{2A, 2B\}, \phi(2B) = \{2A, 2B\}, \phi(2C) = \{2C, 2D\}, \phi(2D) = \{2C, 2D\}$. Since the children of these nodes are all leaf nodes, we need not define the subtree mapping functions $\phi_{2A,2B}, \phi_{2C,2B}$.

2.2. Mapping behavioral strategies from the abstract game back to the original game

When computing a strategy profile σ' in M', we need some notion of how to interpret this profile in M. Here we introduce two classes of mappings from abstract strategies to real strategies. First, we present the natural extension of lifted strategies introduced by Sandholm and Singh [2012] to extensive-form games. In this definition, a valid mapping $\sigma^{\uparrow\sigma'}$ is any strategy profile such that for each information set I and abstract action $a' \in A_{f(I)}$, the probability $\sigma'(f(I), a')$ of selecting that abstract action is divided any way across the actions $g^{-1}(a') \cap I$.

Definition 2.1. (Strategy lifting) Given an abstract game M' and a strategy profile σ', a lifted strategy for M is a strategy profile $\sigma^{\uparrow\sigma'}$ such that for all I, all $a' \in A_{f(I)}$: $\sum_{a \in g^{-1}(a')} \sigma^{\uparrow\sigma'}(I, a) = \sigma'(f(I), a')$.

For our game-theoretic results, we require a stronger notion of similarity in the mapping. We introduce *undivided lifted strategies*, a refinement of lifted strategies. In addition to the lifted strategy constraint, we require that the probability distribution over nodes in an information set $I \in M$ such that $\sigma_{-0}(I) > 0$, when disregarding nature probabilities, conditioned on reaching the information set, is the same as for the abstract information set, when summing over node mappings.

Definition 2.2. (Undivided strategy lifting)) Given an abstract game M' and a strategy profile σ', an undivided lifted strategy for M is a lifted strategy profile σ such that for all $I \in M$ we have

(1) For $s \in I_\phi$: $\frac{\sigma_{-0}(s)}{\sigma_{-0}(I)} = \sum_{s' \in \phi(s)} \frac{\sigma_{-0}(s')}{\sigma'_{-0}(f(I))}$, and
(2) For $s' \in f(I_{\neg\phi})$: $\frac{\sigma'_{-0}(s')}{\sigma'_{-0}(f(I))} = \sum_{s \in h_I^{-1}(s')} \frac{\sigma_{-0}(s)}{\sigma_{-0}(I)}$

One natural family of action abstraction techniques is to select actions for the abstract game from the actions in the original game (as opposed to creating new actions), or, in other words, to simply remove actions from the original game to generate the abstract game. That is, the abstraction is created by restricting play only to a subset of the actions, and potentially also restricting nature to only choosing a subset of its outcomes. Then, the strategy mapping just consists of playing the strategy computed in the restricted game and ignoring the remaining actions. All prior action abstraction algorithms have operated within that family (e.g., [Gilpin et al. 2008; Schnizlein et al. 2009; Hawkin et al. 2011, 2012; Sandholm and Singh 2012; Ganzfried and Sandholm 2013; Brown and Sandholm

2014]). Such lifting (as long as the information abstraction is done within the framework of our paper described above) is a form of undivided strategy lifting.

2.3. Value functions for the original and abstract game

We define value functions both for individual nodes and for information sets.

Definition 2.3. The value for Player i of a given node s under strategy profile σ is

$$V_i^{\sigma}(s) = \sum_{z \in Z_s} \sigma[s \to z] V_i(z)$$

We use the definition of counterfactual value of an information set, introduced by Zinkevich et al. [2007], to reason about the value of an information set under a given strategy profile. The counterfactual value of an information set I is the expected utility of the information set, assuming that all players follow strategy profile σ, except that Player i plays to reach I.

Definition 2.4. The counterfactual value for Player i of a given information set I under strategy profile σ is

$$V_i^{\sigma}(I) = \begin{cases} \sum_{s \in I} \frac{\sigma_{-i}(s)}{\sigma_{-i}(I)} \sum_{z \in Z_s} \sigma[s \to z] V_i(z) & \text{if } \sigma_{-i}(I) > 0 \\ 0 & \text{if } \sigma_{-i}(I) = 0 \end{cases}$$

Analogously, $W_i^{\sigma'} : \mathcal{S}_i' \to \mathbb{R}$ and $W_i^{\sigma'} : \mathcal{I}_i' \to \mathbb{R}$ are the respective functions for the abstract game.

For the information set I_r that contains just the root node r, we have that $V_i^{\sigma}(I_r) = V_i^{\sigma}(r)$, which is the value of playing the game with strategy profile σ. We assume that at the root node it is not nature's turn to move. This is without loss of generality since we can insert dummy player nodes above it.

We show that for information set I at height $k \in \mathcal{H}_i$, $V_i^{\sigma}(I)$ can be written as a sum over descendant information sets at height $\hat{k} \in \mathcal{H}_i$, where $\hat{k}$ is the next level below k that belongs to Player i.

PROPOSITION 2.5. *Let I be an information set at height $k \in \mathcal{H}_i$ such that the there is some $\hat{k} \in \mathcal{H}_i, \hat{k} < k$. For such I, $V_i^{\sigma}(I)$ can be written as a weighted sum over descendant information sets at height $\hat{k}$, and similarly for $W_i^{\sigma'}(I')$:*

$$V_i^{\sigma}(I) = \sum_{a \in A_I} \sigma(I,a) \sum_{\hat{I} \in \mathcal{D}_I^a} \frac{\sigma_{-i}(\hat{I})}{\sigma_{-i}(I)} V_i^{\sigma}(\hat{I})$$

$$W_i^{\sigma'}(I') = \sum_{a' \in A_{I'}} \sigma'(I',a') \sum_{\hat{I}' \in \mathcal{D}_{I'}^{a'}} \frac{\sigma'_{-i}(\hat{I}')}{\sigma'_{-i}(I')} W_i^{\sigma'}(\hat{I}')$$

2.4. Equilibrium concepts

In this section we define the main equilibrium concepts we use. We start with two classic ones.

Definition 2.6 (*Nash equilibrium*). A Nash equilibrium in a game M with root node r is a strategy profile σ such that for each agent i, given σ_{-i}, σ_i is a utility maximizing strategy for i. In other words, for all $i, \bar{\sigma}_i,\ V_i^{\sigma}(r) \geq V_i^{\sigma_{-i},\bar{\sigma}_i}(r)$.

Definition 2.7 (*ϵ-Nash equilibrium*). An ϵ-Nash equilibrium in a game M with root node r is a strategy profile σ such that for all $i, \bar{\sigma}_i,\ V_i^{\sigma}(r) + \epsilon \geq V_i^{\sigma_{-i},\bar{\sigma}_i}(r)$.

Next, we introduce a new equilibrium refinement, *self-trembling equilibrium*. It is a Nash equilibrium where the player assumes that opponents make no mistakes, but she might herself make mistakes, and thus her strategy must be optimal for all information sets that she could mistakenly reach by her own fault.

Definition 2.8 (*Self-trembling equilibrium*). For a game M, a strategy profile σ is a self-trembling equilibrium if it satisfies two conditions. First, it must be a Nash equilibrium. Second, for any information set I belonging to Player i such that $\sigma_{-i}(I) > 0$, σ_i must be a utility maximizing strategy assuming the information set is reached. That is, for all alternative strategies $\bar{\sigma}_i$:

$$V_i^{\sigma}(I) \geq V_i^{\sigma_{-i},\bar{\sigma}_i}(I)$$

We call this second condition the *self-trembling* property.

We introduce this Nash equilibrium refinement because it turns out to be a useful tool for reasoning inductively about quality of Nash equilibria. For perfect-recall games, self-trembling equilibria are a strict superset of perfect Bayesian equilibria. A Perfect Bayesian equilibrium is a strategy profile and a belief system such that the strategies are sequentially rational given the belief system, and the belief system is *consistent*. A belief system is consistent if Bayes' rule is used to compute the belief for any information set where it is applicable. In a perfect recall game, the information sets where Bayes' rule is applicable are exactly the ones where $\sigma_{-i}(I) > 0$. These are exactly the information sets where self-trembling equilibria are sequentially rational. Perfect Bayesian equilibria are a strict subset because there might not exist a belief system for which a self-trembling equilibrium is rational on information sets where Bayes' rule does not apply. In particular, for a given information set where Bayes' rule does not apply, there might be an action that is strictly worse than another action for every node in the information set. Since Bayes' rule does not apply, the self-trembling property places no restrictions on the information set, and thus may play the strictly dominated action with probability one. This is not sequentially rational for *any* belief system.

3. REWARD-APPROXIMATION AND INFORMATION-APPROXIMATION ERROR TERMS

We define approximation error terms, that give us a notion of the distance between the real game and the abstract game. These will later be used to prove bounds on the solution quality of equilibria computed using the abstract game.

First we define *reward approximation error*. Intuitively, we define the reward error at leaf nodes to be the maximum difference between values of the real leaf and the abstract leaves that it maps to, at player nodes to be the maximum error of the child nodes, and at nature nodes to be a weighted sum of the error of the child nodes. Formally, the reward approximation error for the mapping from M to M' for a given agent i, and node s is

$$\epsilon^R_{s,i} = \begin{cases} \max_{s'\in\Phi(s)} |V_i(s) - W_i(s')| & \text{if } s \text{ is a leaf node} \\ \max_{a\in A_s} \epsilon^R_{t^s_a,i} & \text{if } s \text{ is a player node} \\ \sum_{a\in A_s} \sigma_0(s,a)\epsilon^R_{t^s_a,i} & \text{if } s \text{ is a nature node} \end{cases}$$

Here $\Phi^{-1}(s) = \bigcup_{\hat{s}\in\mathcal{S}_\phi, s'\in\phi(\hat{s})} \phi_{\hat{s},s'}(s) \cup h(s)$ is the set of all abstract nodes that s maps to through either some $\phi_{\hat{s},s'}$ function or h. For a given strategy profile, the error for node s for a non-leaf, non-nature node could be defined as the weighted sum of the errors of the actions at the state. But since we do not know which strategy will be played, we have to take the maximum. The reward error for Player i of an information set I is the maximum error over the nodes in I, and the total reward error is the maximum of this quantity over all players:

$$\epsilon^R_{I,i} = \max_{s\in I} \epsilon^R_{s,i}, \quad \epsilon^R = \max_i \epsilon^R_{r,i}$$

Similarly, we define the *distribution error* for an information set I and nodes $s' \in f(I_{\neg\phi})$ and $s \in I_\phi$ to be:

$$\epsilon^D_{I,s'} = \left| \frac{\sum_{s \in h_I^{-1}(s')} \sigma_{-i}(s)}{\sigma(I)} - \frac{\sigma'(s')}{\sigma'(f(I))} \right|, \quad \epsilon^D_{I,s} = \left| \frac{\sum_{s' \in \phi(s)} \sigma'_{-i}(s')}{\sigma'(f(I))} - \frac{\sigma(s)}{\sigma(I)} \right|$$

The distribution error for information set I and error over all information sets at height k is:

$$\epsilon^D_I = \sum_{s' \in f(I)} \epsilon_{I,s'} + \sum_{s \in I_\phi} \epsilon^D_s, \quad \epsilon^D_k = \max_{I \in \mathcal{I}_k} \epsilon^D_I$$

For all distribution error notions, they depend on the strategy profile played, but we omit this in the subscripts for ease of readability. For all proofs, it will be clear what the strategy profile is. We define the nature distribution error of an information set I and nodes $s' \in f(I_{\neg\phi}), s \in I_\phi$ to respectively be

$$\epsilon^0_{I,s'} = \left| \frac{\sum_{s \in h_I^{-1}(s')} \sigma_0(s)}{\sigma_0(I)} - \frac{\sigma'_0(s')}{\sigma'_0(f(I))} \right|, \quad \epsilon^0_{I,s} = \left| \frac{\sum_{s' \in \phi(s)} \sigma'_0(s')}{\sigma'_0(f(I))} - \frac{\sigma_0(s)}{\sigma_0(I)} \right|$$

This is the difference between nature's probability of reaching s' or s and nature's probability of reaching any node in $h_I^{-1}(s')$ and in $\phi(s)$ respectively, all normalized by the probability of reaching the given information sets. The nature error for information set I is

$$\epsilon^0_I = \sum_{s' \in f(I_{\neg\phi})} \epsilon^0_{I,s'} + \sum_{s \in I_\phi} \epsilon^0_s$$

For a nature node s at height $k \in \mathcal{H}_0$ and $s' \in \Phi(s)$, we define the nature action $a' \in A_{s'}$ error and node error to respectively be

$$\epsilon^0_{s,s',a'} = \left| \sigma'(s',a') - \sum_{a \in g^{-1}(a') \cap A_s} \sigma_0(s,a) \right|, \qquad \epsilon^0_s = \max_{s' \in \Phi(s)} \sum_{a' \in A_{s'}} \epsilon^0_{s,s',a'}$$

The nature error at height k is

$$\epsilon^0_k = \begin{cases} \max_{I \in \mathcal{I}_k} \epsilon^0_I & \text{if } k \notin \mathcal{H}_0 \\ \max_{s \in S_k} \epsilon^0_s & \text{if } k \in \mathcal{H}_0 \end{cases}$$

In contrast to distribution error, nature distribution error does not depend on the strategy profile. For any undivided lifted strategy, the distribution error is bounded by the nature distribution error.

PROPOSITION 3.1. *For any undivided lifted strategy σ and information set I, $\epsilon^D_I \le \epsilon^0_I$.*

PROOF. This follows immediately from the definition of an undivided lifted strategy. □

We show that for a given node s and abstract node s' that is a descendant of $h(s)$, the probability of reaching s' from $h(s)$ is at least as high as the probability of reaching any node in $h^{-1}(s')$ from s, disregarding error in nature:

PROPOSITION 3.2. *For any node s, abstract node s' that's a descendant of $h(s)$, and strategy σ lifted from σ',*

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma[s \to \hat{s}] \le \sigma'[h(s) \to s'] + \sum_{l \in \mathcal{H}_0, k \ge l > k'} \epsilon^0_l$$

The same result holds for the set of nodes $\phi^{-1}_{\hat{s},\hat{s}'}(s')$ for all $\hat{s}, \hat{s}'$ such that $\phi_{\hat{s},\hat{s}'}$ is defined.

4. LIFTED STRATEGIES FROM ABSTRACT EQUILIBRIA HAVE BOUNDED REGRET

We are now ready to move to the main part of the paper, which is a mathematical framework for bounding the loss from abstraction in extensive-form games.

We first show that the utility obtained from implementing any lifted strategy in M is close to the utility of the original strategy in M'. We will use this to prove our main result.

PROPOSITION 4.1. *For any player i, abstract strategy σ', and lifted strategy $\sigma^{\uparrow\sigma'}$:*

$$|V_i^{\sigma^{\uparrow\sigma'}}(r) - W_i^{\sigma'}(f(r))| \leq \epsilon_i^R + \sum_{j \in \mathcal{H}_0} \epsilon_j^0 \overline{W}$$

We are now ready to show our main result, which is game theoretic. We show that any Nash equilibrium computed in the abstract game, when implemented as an undivided lifted strategy in the original game, has bounded regret.

THEOREM 4.2. *For any Nash equilibrium σ' in M', any undivided lifted strategy σ is a $2\epsilon^R + \sum_{j\in\mathcal{H}_i} \epsilon_j^0\overline{W} + \sum_{j\in\mathcal{H}_0} 2\epsilon_j^0\overline{W}$-Nash equilibrium in M.*[2]

While this is our main result and the proof is a central contribution of the paper, the proof is quite long, so we relegate it to the appendix due to limited space. Here, we give an overview of the techniques that we use. The proof operates on a per-player basis, since the other players can be considered constant for computing regret for a single player. We inductively show bounds for all information sets belonging to the player, using the counterfactual value to reason about information sets in a bottom-up fashion. Since our induction is on the player's information sets, we introduce several new analysis tools to handle the fact that each step in the induction reasons about rationality across several branches of the game tree. First, we rely on the perfect recall property, which allows us to use the counterfactual value of the information set, with the distribution over the nodes not depending on the player. For information sets with probability zero of being reached, we use the self-trembling property from Definition 2.8. Since this requirement relies only on the player himself for any refinement over Nash equilibria, it allows us to reason about what the player could have received at these information sets, which we need in order to bound his loss at information sets reached with positive probability. We do *not* assume that a self-trembling equilibrium was computed, we only use the fact that the player *could* have played a self-trembling strategy as a tool to bound his regret for any Nash equilibrium. Finally, we need to handle the possibility of information sets from the original game not having the same distribution over nodes as the abstract information set it maps to. By mapping to undivided lifted strategies, we can remove any dependency on the mapped strategies of the other players, and bound the difference in distribution over the nodes in the set entirely by our nature distribution error definitions.

Example where the bound is tight. We now investigate the tightness of our bounds. First, we show that the bound over payoff error is tight.

PROPOSITION 4.3. *There exist games such that requiring any amount of abstraction leads to every Nash equilibrium of the abstraction having regret $2\epsilon^R$ for some player when implemented in the original game with any undivided strategy.*

For the nature error terms, we show that the bound is tight up to a constant factor of 6.

PROPOSITION 4.4. *There exist games such that requiring any amount of abstraction leads to some Nash equilibrium of the abstraction having regret $\frac{1}{6}(\sum_{j\in\mathcal{H}_i} \epsilon_j^0\overline{W} + \sum_{j\in\mathcal{H}_0} 2\epsilon_j^0\overline{W})$ for some player when implemented in the original game with any lifted strategy.*

[2]With the same proof, except without using the self-trembling property for subgame-inducing nodes, we can show that any subgame-perfect equilibrium in the abstract game maps to an approximate subgame-perfect equilibrium in the original game. This requires that subgame-inducding nodes in the real game map to subgame-inducing nodes in the abstract game. This is easily achieved by disallowing multi-mapping on subgame-inducing nodes.

In both the above tightness examples, we relied on either mapping only undivided lifted strategies (in the first case), or showing the bound is tight for only some equilibria (in the latter case). However, computing the best lifted strategy can, in general, amount to computing an equilibrium in the full game, as it does in the first case. Likewise, avoiding certain equilibria in the abstract game would require somehow adding more information to the abstraction computation, thus linking it more intimately to the original game. This was considered by Johanson et al. [2013], but their evaluation requires computing best responses in the original game.

5. ABSTRACTION ALGORITHMS

In this section we develop several algorithms for computing abstractions in extensive form games, such that Nash equilibria computed in these abstractions satisfy the bound derived in Theorem 4.2 when mapped to an undivided lifted strategy in the original game.

5.1. Level-by-level abstraction

Prior game abstraction algorithms typically proceed level by level in the game tree—possibly moving both up and down—e.g. [Gilpin and Sandholm 2006, 2007a,b; Gilpin et al. 2007, 2008; Zinkevich et al. 2007; Sandholm and Singh 2012; Johanson et al. 2013; Ganzfried and Sandholm 2014]. We give a general framework for a large class of level-by-level abstraction algorithms, and investigate the required subroutines. We then present an impossibility result for level-by-level abstraction.

First we give a general framework for computing abstractions level by level. The algorithm operates on a game M, and a set of nodes S at some level k, where S would usually be an information set. It proceeds to build the set A_{Iso}, a set of action pairs that are eligible for merging, along with the cost c of merging them. This is done by iterating over all action pairs a_1, a_2 available at the nodes, and calling the subroutine APPROXIMATELYISOMORPHIC, which computes whether the actions can be merged and at what cost, for each node $s \in S$. Once the set A_{Iso} has been constructed, the algorithm calls a subroutine COMPUTEPROTOTYPES, which selects the subset of actions that are kept (we call these *prototypes*), with the remaining branches mapped onto the prototypes.[3]

ALGORITHM 1: A framework for level-by-level abstraction with bounded loss in extensive-form games.

```
Input: A game $M$, a set of nodes $S$ at some level $k$ such that they have the same action set available
Output: An abstract game $M'$
$A_{Iso} \leftarrow \emptyset$
for each action pair $a_1, a_2 \in A_S$ do
    for each node $s \in S$ do  Test ApproximatelyIsomorphic($t^s_{a_1}, t^s_{a_2}$)
    Let $c$ be the cost of merging $a_1, a_2$ over all $s \in S$
    if isomorphic for all $s \in I$ then add $((a_1, a_2), c)$ to $A_{Iso}$
    Call ComputePrototypes($A_{Iso}$) to compute a set of prototypes that the remaining actions map onto.
```

The two subroutines APPROXIMATELYISOMORPHIC and COMPUTEPROTOTYPES perform the brunt of the work in this algorithm. We will now, in turn, examine what the subroutines entail.

5.1.1. Extensive-form game tree isomorphism. In order to determine whether two given actions can be merged at some information set (APPROXIMATELYISOMORPHIC function above), it must be determined what the cost of merging the subtrees are at each node in the information set. A special, and simple, case is when the subtrees are subgames, and there is only a single node in the information set. Consider such two subgames rooted at nodes s_1, s_2. Since we are assuming that abstraction is conducted level by level, the algorithmic problem amounts to finding the cheapest onto mapping of s_1 onto s_2. To reason about this problem, we introduce the *extensive-form game*

[3]The theory from earlier in this paper also applies to the setting where the abstract game has new actions, if the new action is constructed from an action in the original game (and that subtree under that action) by simply changing payoffs in that subtree. However, as in all prior abstraction algorithms, in this algorithms section we focus on abstraction that selects action prototypes from the actions in the original game.

tree isomorphism problem. For the purposes of this section, we will consider the node mapping function h an onto function, rather than a surjection, since abstraction is not performed as part of this subroutine.

First we introduce a simple notion of isomorphism. It does not capture everything we need for the APPROXIMATELYISOMORPHIC subroutine, but we will later show that even this simple notion is hard to compute. For two game trees to be isomorphic, this definition requires two parts. First, the trees must be isomorphic in the traditional sense. Second, for any node pairs from the same information set, their mapped nodes have to be in the same information set, and for any node pairs in different information sets, their mapped nodes must be in different information sets.

Definition 5.1. (Extensive form game tree topology isomorphism) A topology isomorphism (TI) between extensive form games M, M' is an onto mapping h such that

(1) For nodes $s_1, s_2 \in M$ belonging to the same information set I, $h(s_1)$ and $h(s_2)$ belong to the same information set in M'.
(2) For nodes $s_1 \in I_1$ and $s_2 \in I_2$, where $I_1 \neq I_2$, $h(s_1)$ and $h(s_2)$ belong to different information sets in M'.
(3) For nodes s and s_c, where s_c is a child of s, $h(s_c)$ is a child of $h(s)$.

For any tree isomorphism on two extensive-form game trees (that is, ignoring information sets), any node at level k in M will map to a node $h(k)$ at level k in M'. For leaf nodes, they are the only nodes with out degree one, so leaf nodes must map to leaf nodes. Likewise, the root node r has some distance d to each leaf node, since we assumed uniform depth, so $h(r)$ must map to the root node in M'. Now consider some node s at height k, at depth d. $h(s)$ must be at height k in order to be isomorphic, and the shortest path to the root node must be of length d. Thus, our requirement that a TI must respect player's turn taking is achieved already by it being a tree isomorphism.

We now present a refinement of TI, which we call *extensive form game tree strategic isomorphism (SI)*, that captures the chance and utility structure of the game. This refinement is what we need to compute for the APPROXIMATELYISOMORPHIC subroutine. For subtrees where this isomorphism does not exist, we want to compute the mapping that minimizes the distance to strategic isomorphism.

Definition 5.2. (Extensive form game tree strategic isomorphism) An extensive form game tree strategic isomorphism (SI) is a TI that satisfies the following further constraints.

(1) For all nodes s at some height $k \in \mathcal{H}_0$, for each $a \in A_s$, $\sigma(s, a) = \sigma(h(s), g(a))$.
(2) For all leaf nodes $z \in Z$ and players $i \in N$, $V_i(z) = V_i(h(z))$.

We now show that both TI and SI are graph isomorphism-complete.

THEOREM 5.3. *Deciding whether two game trees are extensive-form game tree topologically isomorphic or extensive-form game tree strategically isomorphic is graph isomorphism-complete.*

A graph isomorphism complete problem is a problem such that there exists a polynomial time reduction both to and from the graph isomorphism problem [Booth and Colbourn 1979]. Game isomorphism has been studied to a limited extent in the literature. The more general question of whether two games are isomorphic across game types has been studied by Gabarró et al. [2007]. Peleg et al. [1999] and Sudhölter et al. [2000] study extensive form game tree isomorphism in the context of strategic equivalence, but do not consider the computational problem. To our knowledge, ours is the first paper to introduce the question of computing extensive-form game tree isomorphisms.

5.1.2. Choosing the set of prototypes. After computing the loss incurred by merging the different action pairs available at the set of nodes under consideration, the subroutine COMPUTEPROTOTYPES computes a subset of the action pairs whose subtrees are kept as the abstract game, with the remaining subtrees mapped onto these. We now show that, even when the values of merging all action pairs have been computed, the problem of choosing the optimal subset of actions is NP-complete.

Definition 5.4. (Action subset selection problem) The input is a set of actions A, a cost c_{a_1,a_2} for merging each action pair $a_1, a_2 \in A$, some value k that specifies the number of actions to be selected, and a cost c. The action subset selection problem asks if there exists a set of k actions and a mapping of the remaining actions onto this set, such that the cost is less than or equal to c.

THEOREM 5.5. *The action subset selection problem is NP-complete.*[4]

The problem of computing an extensive form game abstraction that minimizes the bound given in Theorem 4.2 is easily seen to be NP-complete as well—even with just one player, and thus it holds for zero-sum games also. The action subset selection problem itself reduces to a two-level game tree by making a chance node as the root, with all children being in the same information set of size $|A|$, with $|A|$ actions available, where each node is used to ensure the correct cost of merging for a single action pair, with all other merges being free at the node.

5.1.3. A sequence of single abstraction transformations to the optimal abstraction might not exist. We now show a fundamental issue with level-by-level abstraction. It can leave valid (even lossless) abstractions undiscovered: reasoning across merges at different levels and in different subtrees simultaneously is required in order to stay within the set of valid abstractions throughout the traversal of the game tree. To show this, we show a slightly broader result: determining whether a merge is within the desired error bound can require reasoning about merges in subtrees different from the ones considered for merging.

THEOREM 5.6. *Any abstraction technique that computes the abstraction by merging subtrees must satisfy at least one of the following properties.*

(*1*) *It does not always find the smallest abstraction in the space of valid abstractions for a given bound.*
(*2*) *When merging two nodes at a level, it must ensure that future merges at other levels in the tree, outside of the two subtrees below those two nodes, will enable returning to an abstraction within the bound.*

This applies even if the game is a game of ordered signals [Gilpin and Sandholm 2007b], zero-sum, and only information abstraction is performed.

PROOF. Consider a two-player poker game where the card deck consists of two kings and two jacks. Each player is dealt a private card and then a single public card is dealt. We consider two variants based on what the payoffs at the leaves are. In the first variant, Player 1 always wins. The second variant is a slight modification, where if Player 2 pairs a private $J2$ with a public $K2$ she wins, and otherwise Player 1 wins. Both variants are zero-sum games of ordered signals. Figure 2 shows the possible sequences of cards dealt. The two variants are obtained by setting δ equal to 1 or -1 in Figure 2, respectively. If Player 1 always wins ($\delta = 1$), the whole game can be abstracted into a single string of cards dealt, as none of them matter. If Player 2 wins ($\delta = -1$), the two public kings cannot be abstracted into a single king when Player 1 holds $J1$ and Player 2 holds $J2$ based on the following reasoning. Consider $K1$ and $K2$ dealt to Player 1 at the root. Player 2 has two information sets that consist of him holding a $J2$ and the public card being a king. Each of those two information sets has two nodes corresponding to Player 1 holding the other jack or the other king. The two information sets differ based on whether the public king is $K1$ or $K2$. If $K1$ and $K2$ at the root are abstracted into a single tree, then one of these two information sets loses a node, since the sequence $K1, J2, K2$ would map onto $K2, J2, K1$, or vice versa. This leads to Player 2

[4]Sandholm and Singh [2012] show that in stochastic games it is NP-complete to compute an abstraction where one of the players does not automatically receive the worst possible payoff. However, some bound-minimizing solutions do not solve the problem that is reduced from. Thus that result does not imply that minimizing the bound is NP-complete.

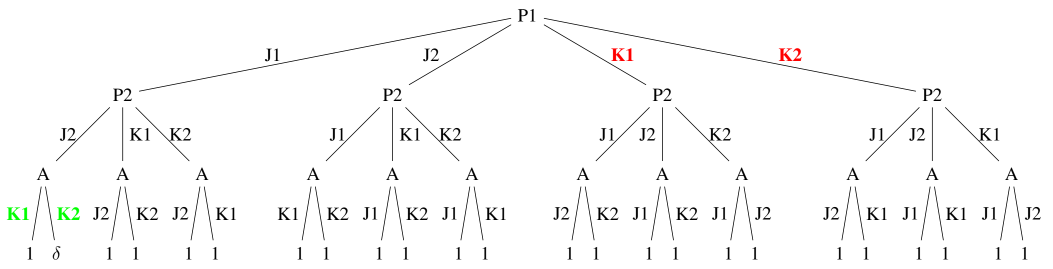


Fig. 2. A signal tree for a simple poker game. Nodes labeled P1 or P2 denote the card being dealt privately to player 1 or 2, respectively. Nodes labeled A denote a public card. A leaf node labeled 1 indicates Player 1 winning.

being able to deduce exactly which card Player 1 has for the information set that is now a singleton. For the other information set, the probability on the node where Player 1 has a private king would be higher than the probability on the node where Player 1 has a private jack. Thus, if a lossless abstraction is desired, the choice of whether to abstract $K1$ and $K2$ at the root hinges on whether δ is set to 1 or not. $\square$

One consequence is that the GameShrink lossless abstraction algorithm [Gilpin and Sandholm 2007b] does not work correctly for all games of ordered signals.

5.2. Generating an abstraction by considering all levels at once

Motivated by the problems described in Section 5.1, we develop an algorithm for computing abstractions with bounded loss that operates on all levels at once. The only assumption we make about structure of the game, is that we only allow nature nodes to merge if they have the same number of actions, and only allow different branches to merge if they have the same probability of occurring.

Using integer-programming (IP), we develop two similar models for computing abstractions. One takes as input the maximum number of game tree nodes allowed and computes an abstraction that minimizes the bound given in Theorem 4.2. The other takes as input a desired error bound and computes the smallest abstraction subject to satisfying the bound.

5.2.1. Variables. For each node s_i, we introduce a variable $p_i \in \{0,1\}$ denoting whether s_i is a prototype. For each level $k \in \mathcal{H}$ and ordered pair of nodes at height k, $s_i, s_j \in S_k$, we introduce a variable $\delta_{i,j} \in \{0,1\}$ denoting whether s_i is mapped onto s_j. For each unordered pair of information sets I_i, I_j at height k, we introduce a variable $I_{i,j} \in \{0,1\}$ denoting whether the two information sets map to the same abstract information set.

5.2.2. Objective functions. In the case where we are given a bound to satisfy and wish to compute the smallest abstraction, we maximize the sum over all abstraction variables $\delta_{i,j}$, thereby minimizing the number of nodes in the game tree:

$$\max_{\delta_{i,j},p_i,I_{i,j}} \sum_{i,j} \delta_{i,j} \tag{1}$$

In the case where we are given a maximum tree size and want to minimize the bound, we minimize the sum over leaf nodes mapped onto each other, weighted by the absolute difference in utility at the leaves:

$$\min_{\delta_{i,j},p_i,I_{i,j}} \sum_{z\in Z}\sum_{\hat{z}\in Z} \max_{i\in N} |V_i(z) - V_i(\hat{z})|\delta_{i,j} \tag{2}$$

5.2.3. Constraints. To ensure that nodes are either prototypes or mapped onto a single prototype, we introduce the following three constraints. The first and second constraints below are introduced for all nodes s_i, and the third for all pairs of nodes s_i, s_j at the same height.

$$\sum_{j \in S_k} \delta_{i,j} \leq 1, \forall s_i \in S, \qquad p_i + \sum_{j \in S_k} \delta_{i,j} \geq 1, \forall s_i \in S, \qquad \delta_{j,i} - p_i \leq 0, \forall k \in \mathcal{H}, s_i, s_j \in S_k \tag{3}$$

The first constraint ensures that each node is mapped onto at most one other node. The second and third constraints ensure that the variables p_i accurately reflect whether s_i is a prototype. The second constraint requires that $p_i = 1$ unless s_i is mapped to some other node. The third constraint sets $p_i = 0$ if s_i is mapped onto any other node s_j. Together, the last two constraints ensure that nodes are only mapped onto prototypes, and that prototype nodes are not mapped onto anything.

If a node s_i is mapped onto another node s_j, one of the constraints specified in Section 2.1 requires that the children at s_i map onto the children at s_j. This is ensured by the following constraint, where C_i is the set of all child node indices of s_i.

$$\delta_{i,j} - \sum_{c_j \in C_j} s_{c_i,c_j} \leq 0, \forall s_i \in S \setminus Z, c_i \in C_i, \tag{4}$$

If $\delta_{i,j} = 1$, this constraint requires that s_{c_i} is mapped onto some child node of s_j by requiring that at least one of them is set to one. Similarly, if s_i is mapped onto s_j, we require that the parent node of s_i, denoted by s_{p_i}, is mapped onto the parent node of s_j, denoted by s_{p_j}. This gives the following constraint, where the parent mapping variable δ_{p_i,p_j} is required to be set to one if $\delta_{i,j} = 1$.

$$\delta_{i,j} - \delta_{p_i,p_j} \leq 0, \forall k \in \mathcal{H}, s_i, s_j \in S_k \tag{5}$$

For each node pair s_i, s_j at some height k, let I_i, I_j be their respective information sets and $I_{i,j}$ the variable denoting whether the information sets are merged. If the nodes are merged, we require that their information sets are also merged, which is achieved by the following constraint.

$$\delta_{i,j} - I_{i,j} \leq 0, \forall k \in \mathcal{H}, s_i, s_j \in S_k \tag{6}$$

Information set merges are transitive, so if two information sets are both merged with the same third information set, we require that they are themselves merged:

$$I_{i,j} + I_{i,l} - I_{j,l} \leq 1, \forall k \in \mathcal{H}, I_i, I_j, I_l \in \mathcal{I}_k \tag{7}$$

Using the variable $I_{i,j}$ for two information sets I_i, I_j, we can ensure that each prototype node in the abstract merged information set has a node from each information set mapping onto it. Without loss of generality, assume $s_l \in I_i$, we add the following constraint.

$$p_l + I_{i,j} - \sum_{s_m \in I_j} \delta_{m,l} \leq 1, \forall I_i, I_j, s_l \in I_i \tag{8}$$

This requires that if s_l is a prototype, $p_l = 1$, and if $I_{i,j} = 1$, at least one node from I_j maps onto s_l.

As mentioned above, we only allow nature outcomes to map onto each other if their probability is the same. Further, if for some nature node s_j mapped onto a nature node s_i we have that $m > 1$ child nodes of s_j map onto the same child node of s_i, then we must ensure that $m - 1$ child nodes at s_i map onto c_i, in order to keep the nature distribution error at zero. This is achieved by the following two constraints.

$$\delta_{c_i,c_j} = 0, \forall s_i, s_j, c_i \in C_i, c_j \in C_j, \sigma(s_i, c_i) \neq \sigma(s_j, c_j) \tag{9}$$

$$\sum_{c_j \in C_j} \delta_{c_j,c_i} = \sum_{c_k \in C_i} \delta_{c_k,c_i} + 1, \forall s_i, s_j \in S \setminus Z, c_i \in C_i \tag{10}$$

The first constraint just disallows mapping children of nature nodes that do not have equal probability of occurring. The second constraint ensures that the probability of a prototype child being chosen at the nature node, which is equal to the sum of the probabilities of outcomes at the node that are mapped to it, is equal to the sum of probabilities over outcomes mapped to it from s_j.

For the case where a specific bound is given as input and we wish to compute the minimum size abstraction, we add the following constraint.

$$\sum_{z_i, z_j \in Z} \max_{i \in N} |V_i(z_i) - V_i(\hat{z_j})| \delta_{i,j} \leq \epsilon^R \tag{11}$$

This constraint sums over all leaf node pair mappings, and weights them by the difference in utility at the pair. We require that this sum be less than ϵ^R, the given bound. For the case where a maximum tree size K is given as input and we wish to minimize the bound, we add the following constraint.

$$\max_{\delta_{i,j}, p_i, I_{i,j}} \sum_{i,j} \delta_{i,j} \geq |S| - K \tag{12}$$

This constraint requires that at least $|S| - K$ merges are performed, so the abstraction has at most K nodes.

The number of variables in the model is $O(|Z|^2)$. The largest constraint sets are those in Equation 7 and those over all variable pairs at each level. The former is $O(\max_{k \in \mathcal{H}} \mathcal{I}_k^3)$ and the latter is $O(|Z|^2)$.

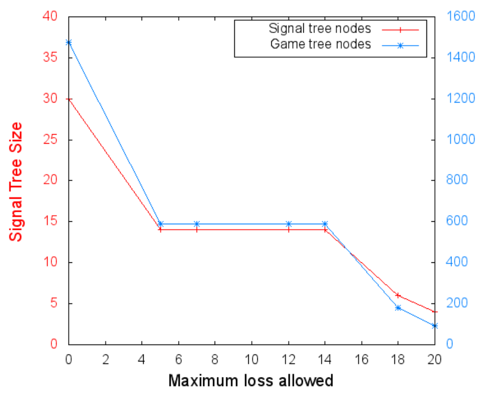
5.2.4. Experiment. We applied our IP model to a simple poker game. Our game has a deck of five cards: two jacks, a queen, and two kings. Two players play the game, and after each round of cards is dealt, up to two rounds of betting occur. A full description of the game is given in the appendices.

One advantage of poker games for testing our approach is that the chance outcomes can be decoupled from the player actions using the *signal tree*. The signal tree is defined conceptually by removing all player actions from the game tree, and only considering the tree of possible nature moves (aka signals). Clearly, for this decoupling to be possible, the nature moves can be conditioned only on which prior nature events have occurred, not on player actions. Any abstraction computed on the signal tree can easily be converted to an abstraction of the full game. Gilpin and Sandholm [2007b] introduced the signal tree in the specific setting of games of ordered signals, a game class closely resembling poker. More generally, in any game where the player's action options are independent of nature's moves, abstraction can be performed on the signal tree.

In our poker game, the unabstracted signal tree has 86 nodes; the game tree has 4806 nodes. The IP has 4,427 binary variables (one for each pair of nodes at each level of the signal tree, plus the additional bookkeeping ones) and 38,813 constraints. To solve the IP, we used CPLEX version 12.5.

For the model where a bound is given as input and the objective is to minimize tree size, we ran experiments with a bound ranging from 0 to 20 chips. Figure 3 Left shows a plot of the game sizes as a function of the bound. As can be seen, tree size is a step function of the given bound. Four different abstraction sizes were found. The coarsest abstraction has four signal tree nodes, and thus represents a single sequence of outcomes where the players act essentially without seeing any cards. The coarsest lossless abstraction has size 30. It is not until we allow a loss bound of 5 that the algorithm finds a lossy abstraction (of size 14). For the model where a maximum tree size is given as input and the objective is to minimize the regret bound, we ran experiments for signal tree size bounds from 4 to 54. Figure 3 Right shows exploitability as a function of allowed signal tree size. Three plots are shown, the exploitability of each of the two players, and the exploitability bound given by Theorem 4.2. By and large, the three plots decrease with signal tree size, with a non-monotonicity at size 6, where Player 1's exploitability goes up compared to that at size 4. This is an instance of abstraction pathology, which exists in games: refining an abstraction can cause the equilibrium strategies to have higher exploitability in the original game [Waugh et al. 2009a].

The different abstractions (signal trees) that the IP generated are presented in the appendix. Interestingly, sometimes the IP generated an abstraction that is asymmetric between the players.



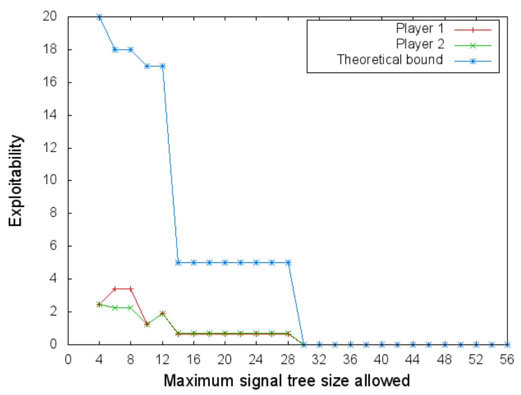


Fig. 3. Left: Number of nodes in the signal tree (left y-axis) and in the game tree (right y-axis) as a function of the allowed loss in the IP model when minimizing tree size. Right: Exploitability as a function of the allowed signal tree size. Exploitability for both players is shown, along with our theoretical bound.

6. DISCUSSION

We presented the first framework for giving theoretical guarantees on solution quality in lossy abstraction in extensive-form games. We defined notions of payoff and nature error between the abstraction and the original game, and developed analysis tools that enable one to give an upper bound on regret and exploitability of equilibria computed in the abstract game, when mapped to the original game. To do this, we introduced a notion of strategy mappings, undivided lifted strategies. We also introduced an equilibrium refinement, self-trembling equilibrium, that we used to analyze the quality of general Nash equilibria from abstract games.

Using this framework, we developed the first lossy extensive-form game abstraction algorithm with bounds on solution quality. We did this by designing an IP that computes either the minimal size abstraction given a bound on solution quality, or a solution with minimum bound given a cap on tree size. Experiments on a relatively simple poker game showed that our method finds a lossless abstraction when one is available and lossy abstractions when smaller abstractions are desired.

While our framework can be used for lossy abstraction, it is also a powerful tool for lossless abstraction if we set the bound to zero. This is, to our knowledge, the first framework to provide tools for characterizing lossless abstractions in general extensive-form games of perfect recall. Likewise, our IP model is the first algorithm to compute abstractions with such guarantees. It is also the first algorithm to automatically detect the canonical representation of poker games while guaranteeing losslessness, without resorting to any domain-specific tricks. Until now, there has been no theory for algorithmically detecting such symmetries in a general sense that goes beyond poker. Further, as we described in Section 5.1.3, the only similar work with theoretical guarantees [Gilpin and Sandholm 2007b] has the problem that it might not be able to reach the lossless abstractions due to discontinuity between the regions of lossless abstractions in the space of all abstractions. This is in spite of their work analyzing a much more restricted game setting, that very closely resembles poker, whereas our theoretical work makes no assumptions about the game beyond perfect recall.

As mentioned in Section 1, our quality results for lossy abstraction generalize similar results for the special case of stochastic games (expanded to a DAG) [Sandholm and Singh 2012]. They get a cubic dependence on the depth of the game tree in their bounds. In our work, if we consider the sum over error terms in the bound in Theorem 4.2, we have $2\epsilon^R + \sum_{j\in\mathcal{H}_i} \epsilon_j^0 \overline{W} + \sum_{j\in\mathcal{H}_0} 2\epsilon_j^0 \overline{W}$. The two expressions $\sum_{j\in\mathcal{H}_i} \epsilon_j^0 \overline{W}$ and $\sum_{j\in\mathcal{H}_0} 2\epsilon_j^0 \overline{W}$ have the biggest dependence on depth in our bound. In converting the stochastic game, $\overline{W}$ gets a linear dependence on the depth, and the summations are linear in the depth. The error term ϵ_j^0 has no dependence on depth for either term, since each state of the stochastic game induces a subgame in the extensive-form representation. Our bound, when applied to a stochastic game DAG thus has only a quadratic dependence on depth, while theirs was cubic.

We introduced the extensive form game tree isomorphism and action subset selection problems, both important for computing abstractions on a level-by-level basis. We proved that the former is graph isomorphism complete, and the latter NP-complete. We showed that level-by-level abstraction can be too myopic and thus fail to find even obvious lossless abstractions. At the same time, it can be challenging to scale up the IP that considers all levels at once, depending on the game.

The hardness results do not mean that abstraction is not helpful. For one, the worst-case hardness results do not imply hardness in practice. Furthermore, many games can be decomposed in terms of information structure. Our algorithm can be used to compute abstractions on the decomposed parts, leading to a smaller game size in the full game. One example of such decomposition is conducting abstraction on the signal tree instead of the full game tree. In fact, essentially all information abstraction algorithms for poker proceed level by level in tree data structures that are like the signal tree, except that each tree consists of signals to only one player. Such algorithms have proven to be key for being able to address large games even if the abstraction algorithm solves NP-complete subproblems when dealing with each level in turn [Gilpin and Sandholm 2007a], even in two-player zero-sum games which are solvable in worst-case polynomial time.

There are many possible future directions for developing abstraction algorithms within our framework that provides bounds, for example, algorithms that proceed level by level but do not necessarily find an optimal abstraction (and may not even optimize within a level), or algorithms that consider multiple levels at once, but potentially not all levels and potentially not optimally.

While our algorithms work for any game of perfect recall, the set of abstractions they can compute is only a subset of all possible abstractions. Recent progress on abstraction for the specific application of solving poker games has been focused on the use of imperfect-recall abstractions [Waugh et al. 2009b; Johanson et al. 2013]. They happen to work well for poker, but are poorly understood. Other practical abstraction algorithms that fall outside our framework are ones that merge different information sets at different branches in the tree without requiring that nodes in the subtrees map to each other, nor that they are in the same information set for the other players. It is easy to construct examples where such abstraction has arbitrarily high loss, but it seems to work well in practice—at least for poker. It would be interesting to explore whether our bounding framework could be extended to these kinds of abstractions.

REFERENCES

BASILICO, N. AND GATTI, N. 2011. Automated abstractions for patrolling security games. In *AAAI Conference on Artificial Intelligence (AAAI).*

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 *International Joint Conference on Artificial Intelligence (IJCAI).*

BOOTH, K. S. AND COLBOURN, C. J. 1979. *Problems polynomially equivalent to graph isomorphism.* Computer Science Department, Univ.

BROWN, N. AND SANDHOLM, T. 2014. Regret transfer and parameter optimization. In *AAAI Conference on Artificial Intelligence (AAAI).*

GABARRÓ, J., GARCÍA, A., AND SERNA, M. 2007. On the complexity of game isomorphism. In *Mathematical Foundations of Computer Science 2007.*

GANZFRIED, S. AND SANDHOLM, T. 2013. Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping. In *International Joint Conference on Artificial Intelligence (IJCAI).*

GANZFRIED, S. AND SANDHOLM, T. 2014. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *AAAI Conference on Artificial Intelligence (AAAI).*

GILPIN, A. AND SANDHOLM, T. 2006. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *National Conference on Artificial Intelligence (AAAI).*

GILPIN, A. AND SANDHOLM, T. 2007a. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

GILPIN, A. AND SANDHOLM, T. 2007b. Lossless abstraction of imperfect information games. *Journal of the ACM 54,* 5. Early version 'Finding equilibria in large sequential games of imperfect information' appeared in ACM Conference on Electronic Commerce (EC), 2006.

GILPIN, A., SANDHOLM, T., AND SØRENSEN, T. B. 2007. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In *AAAI Conference on Artificial Intelligence (AAAI)*.

GILPIN, A., SANDHOLM, T., AND SØRENSEN, T. B. 2008. A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

HAWKIN, J., HOLTE, R., AND SZAFRON, D. 2011. Automated action abstraction of imperfect information extensive-form games. In *AAAI Conference on Artificial Intelligence (AAAI)*.

HAWKIN, J., HOLTE, R., AND SZAFRON, D. 2012. Using sliding windows to generate action abstractions in extensive-form games. In *AAAI Conference on Artificial Intelligence (AAAI)*.

JIANG, A. AND LEYTON-BROWN, K. 2011. Polynomial-time computation of exact correlated equilibrium in compact games. In *ACM Conference on Electronic Commerce (EC)*.

JOHANSON, M., BURCH, N., VALENZANO, R., AND BOWLING, M. 2013. Evaluating state-space abstractions in extensive-form games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

KOLLER, D., MEGIDDO, N., AND VON STENGEL, B. 1996. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior 14,* 2.

LANCTOT, M., GIBSON, R., BURCH, N., ZINKEVICH, M., AND BOWLING, M. 2012. No-regret learning in extensive-form games with imperfect recall. In *International Conference on Machine Learning (ICML)*.

LIPTON, R., MARKAKIS, E., AND MEHTA, A. 2003. Playing large games using simple strategies. In *ACM Conference on Electronic Commerce (ACM-EC)*.

LITTMAN, M. AND STONE, P. 2003. A polynomial-time Nash equilibrium algorithm for repeated games. In *ACM Conference on Electronic Commerce (ACM-EC)*.

PELEG, B., ROSENMÜLLER, J., AND SUDHÖLTER, P. 1999. *The canonical extensive form of a game form: Symmetries*.

SANDHOLM, T. 2010. The state of solving large incomplete-information games, and application to poker. *AI Magazine*. Special issue on Algorithmic Game Theory.

SANDHOLM, T. AND SINGH, S. 2012. Lossy stochastic game abstraction with bounds. In *ACM Conference on Electronic Commerce (EC)*.

SCHNIZLEIN, D., BOWLING, M., AND SZAFRON, D. 2009. Probabilistic state translation in extensive games with large action sets. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

SHI, J. AND LITTMAN, M. 2002. Abstraction methods for game theoretic poker. In *CG '00: Revised Papers from the Second International Conference on Computers and Games*.

SUDHÖLTER, P., ROSENMÜLLER, J., AND PELEG, B. 2000. The canonical extensive form of a game form: Part II. representation. *Journal of Mathematical Economics 33,* 3.

WAUGH, K., SCHNIZLEIN, D., BOWLING, M., AND SZAFRON, D. 2009a. Abstraction pathologies in extensive games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

WAUGH, K., ZINKEVICH, M., JOHANSON, M., KAN, M., SCHNIZLEIN, D., AND BOWLING, M. 2009b. A practical use of imperfect recall. In *Symposium on Abstraction, Reformulation and Approximation (SARA)*.

ZINKEVICH, M., BOWLING, M., JOHANSON, M., AND PICCIONE, C. 2007. Regret minimization in games with incomplete information. In *Conference on Neural Information Processing Systems (NIPS)*.

Online Appendix to: Extensive-Form Game Abstraction With Bounds

CHRISTIAN KROER, Carnegie Mellon University
TUOMAS SANDHOLM, Carnegie Mellon University

Proof of Proposition 2.5

PROOF. From Definition 2.4 and the fact that $\sigma[s \to z] = \sigma(I,a)\sigma[t_a^s \to z]$, we get

$$V_i^\sigma(I) = \sum_{a \in A_I} \sigma(I,a) \sum_{s \in I} \frac{\sigma_{-i}(s)}{\sigma_{-i}(I)} \sum_{z \in Z_{t_a^s}} \sigma[t_a^s \to z] V_i(z)$$

We can write the set $Z_{t_a^s}$ as the union $\bigcup_{\hat{I} \in \mathcal{D}_I^a} \bigcup_{\hat{a} \in A_{\hat{I}}} \bigcup_{\hat{s} \in \hat{I} \cap \mathcal{P}_s} Z_{t_{\hat{a}}^{\hat{s}}}$, i.e. the union over leaf nodes belonging to each of the directly descendant informations sets belonging to the player. In perfect recall games, every $z \in Z_{t_a^s}$ appears in only one of the sets $Z_{t_{\hat{a}}^{\hat{s}}}$. In imperfect recall games, for any z that appears more than once, the probability of the individual paths from t_a^s to z sum up to $\sigma[t_a^s \to z]$. Applying this, we get

$$\begin{aligned}
&\sum_{a \in A_I} \sigma(I,a) \sum_{s \in I} \frac{\sigma_{-i}(s)}{\sigma_{-i}(I)} \sum_{\hat{I} \in \mathcal{D}_I^a} \sum_{\hat{a} \in \hat{I}} \sigma(\hat{I},\hat{a}) \sum_{\hat{s} \in \hat{I} \cap \mathcal{P}_s} \sigma[t_a^s \to \hat{s}] \sum_{\hat{z} \in Z_{t_{\hat{a}}^{\hat{s}}}} \sigma[t_{\hat{a}}^{\hat{s}} \to \hat{z}] V_i(\hat{z}) \\
= &\sum_{a \in A_I} \sigma(I,a) \sum_{\hat{I} \in \mathcal{D}_I^a} \sum_{\hat{a} \in \hat{I}} \sigma(\hat{I},\hat{a}) \sum_{s \in I} \frac{\sigma_{-i}(s)}{\sigma_{-i}(I)} \sum_{\hat{s} \in \hat{I} \cap \mathcal{P}_s} \sigma[t_a^s \to \hat{s}] \sum_{\hat{z} \in Z_{t_{\hat{a}}^{\hat{s}}}} \sigma[t_{\hat{a}}^{\hat{s}} \to \hat{z}] V_i(\hat{z})
\end{aligned}$$

The second line is obtained by rearranging terms. Since i has no active information sets on the path from t_a^s to $\hat{s}$, we have $\sigma_{-i}(s)\sigma[t_a^s \to \hat{s}] = \sigma_{-i}(\hat{s})$, we apply this to get

$$\sum_{a \in A_I} \sigma(I,a) \sum_{\hat{I} \in \mathcal{D}_I^a} \sum_{\hat{a} \in \hat{I}} \sigma(\hat{I},\hat{a}) \sum_{s \in I} \sum_{\hat{s} \in \hat{I} \cap \mathcal{P}_s} \frac{1}{\sigma_{-i}(I)} \sigma_{-i}(\hat{s}) \sum_{\hat{z} \in Z_{t_{\hat{a}}^{\hat{s}}}} \sigma[t_{\hat{a}}^{\hat{s}} \to \hat{z}] V_i(\hat{z})$$

We have that $\frac{1}{\sigma_{-i}(I)} \sigma_{-i}(\hat{s}) = \frac{1}{\sigma_{-i}(I)} \cdot \frac{\sigma_{-i}(\hat{I})}{\sigma_{-i}(\hat{I})} \sigma_{-i}(\hat{s})$, we apply this and rearrange terms to get

$$\begin{aligned}
&= \sum_{a \in A_I} \sigma(I,a) \sum_{\hat{I} \in \mathcal{D}_I^a} \frac{\sigma_{-i}(\hat{I})}{\sigma_{-i}(I)} \sum_{\hat{a} \in \hat{I}} \sigma(\hat{I},\hat{a}) \sum_{s \in I \cap \mathcal{P}_{\hat{I}}} \sum_{\hat{s} \in \hat{I} \cap \mathcal{P}_s} \frac{\sigma_{-i}(\hat{s})}{\sigma_{-i}(\hat{I})} \sum_{\hat{z} \in Z_{t_{\hat{a}}^{\hat{s}}}} \sigma[t_{\hat{a}}^{\hat{s}} \to \hat{z}] V_i(\hat{z}) \\
&= \sum_{a \in A_I} \sigma(I,a) \sum_{\hat{I} \in \mathcal{D}_I^a} \frac{\sigma_{-i}(\hat{I})}{\sigma_{-i}(I)} V_i^\sigma(\hat{I})
\end{aligned}$$

Copyright © 2013 ACM 978-1-4503-2565-3/14/06...$15.00
DOI 10.1145/2600057.2602905 http://doi.acm.org/10.1145/2600057.2602905

Where the last equality follows from Definition 2.4. Similarly, we can show that $W_i^{\sigma'}(I') = \sum_{a' \in A_{I'}} \sigma'(I', a') \sum_{\hat{I}' \in \mathcal{D}_{I'}^{a'}} \frac{\sum_{\hat{s'} \in \hat{I}'} \sigma_{-i}(\hat{s'})}{\sum_{s' \in I'} \sigma'_{-i}(s')} W_i^{\sigma'}(\hat{I}')$. □

Proof of Proposition 3.2

PROOF. We prove this by induction. First, we note that for any $\hat{s} \in h^{-1}(s')$ that's not a descendant of s, $\sigma[s \to \hat{s}] = 0$, so we can ignore such nodes. Base case: Assume that we're given nodes s, s' such that all descendants of s in $h^{-1}(s')$ are children of s and s is at height k. Let I be the information set s belongs to. We get that

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma[s \to \hat{s}] = \sum_{\hat{s} \in h^{-1}(s')} \sigma(I, a_{\hat{s}})$$

where $a_{\hat{s}}$ is the action at s that leads to $\hat{s}$. Definition 2.1 gives

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma(I, a_{\hat{s}}) \leq \sigma'(f(I), a_{s'}) \leq \sigma'[h(s) \to s']$$

If $k \in \mathcal{H}_0$ then the definition of nature error at a node gives:

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma(I, a_{\hat{s}}) \leq \sigma'(f(I), a_{s'}) + \epsilon_k^0 \leq \sigma'[h(s) \to s'] + \epsilon_k^0$$

For the inductive step, assume that for all s, s' such that the length path from s to any $\hat{s} \in h^{-1}(s')$ is below y

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma[s \to \hat{s}] \leq \sigma'[h(s) \to s'] + \sum_{l \in \mathcal{H}_0, k \geq l > k'} \epsilon_l^0$$

Given s, s' such that the path is of length y, we have

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma[s \to \hat{s}] = \sum_{\hat{s} \in h^{-1}(s')} \sigma(I, a_{\hat{s}}) \sigma[t_{a_{\hat{s}}}^s \to \hat{s}]$$

where $a_{\hat{s}}$ is the action taken at s on the path to $\hat{s}$. By the inductive assumption

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma(I, a_{\hat{s}}) \sigma[t_{a_{\hat{s}}}^s \to \hat{s}] \leq \sum_{\hat{s} \in h^{-1}(s')} \sigma(I, a_{\hat{s}}) \sigma[t_{g(a_{\hat{s}})}^{h(s)} \to s'] + \sum_{l \in \mathcal{H}_0, k-1 \geq l > k'} \epsilon_l^0$$

We can then apply exactly the same trick as for the base depending on whether s is a nature node or not, and get

$$\sum_{\hat{s} \in h^{-1}(s')} \sigma(I, a_{\hat{s}}) \sigma[t_{g(a_{\hat{s}})}^{h(s)} \to s'] + \sum_{l \in \mathcal{H}_0, k-1 \geq l > k'} \epsilon_l^0 \leq \sigma'[h(s) \to s'] + \sum_{l \in \mathcal{H}_0, k \geq l > k'} \epsilon_l^0$$

□

Proof of Proposition 4.1

PROOF. We prove this by induction. Since strategic behavior is not considered here, we can ignore information sets and treat each node separately.

Base case: Consider any node n such that all its children are leaf nodes. The utility for this node in the original game, when playing a lifted strategy is:

$$\begin{aligned}
V_i^{\sigma^{\uparrow\sigma'}}(s) &= \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a) V_i(t_a^s) \\
&\leq \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a)(W_i(t_{g(a)}^{h(s)}) + \epsilon_{t_a^s i}^R) \\
&\leq \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a) W_i(t_{g(a)}^{h(s)}) + \epsilon_{s,i}^R \\
&= \sum_{a'\in A_{h(s)}} \sum_{a\in g^{-1}(a')} \sigma^{\uparrow\sigma'}(s,a) W_i(t_{g(a)}^{h(s)}) + \epsilon_{s,i}^R \\
&= \sum_{a'\in A_{h(s)}} \sigma'(h(s),a') W_i(t_{g(a)}^{h(s)}) + \epsilon_{s,i}^R; \text{ by Definition 2.1} \\
&= W_i^{\sigma'}(h(s)) + \epsilon_{s,i}^R
\end{aligned}$$

$V_i^{\sigma^{\uparrow\sigma'}}(s) \geq W_i^{\sigma'}(h(s)) - \epsilon_{s,i}$ is similarly derived, and we will do this direction in the inductive step, for which the technique is very similar. Inductive step: Assume that for any node s with depth less than k: $|V_i^{\sigma^{\uparrow\sigma'}}(s) - W_i^{\sigma'}(h(s))| \leq \epsilon_{s,i} + \sum_{j\in\mathcal{H}_0^{k-1}} \epsilon_j^0 \overline{W}$. We show that it then also holds for nodes of depth k. Assume that we're given a node s of depth k. For a node where the active player is not nature, the utility in the original game, when playing a lifted strategy is:

$$\begin{aligned}
V_i^{\sigma^{\uparrow\sigma'}}(s) &= \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a) V_i^{\uparrow\sigma'}(t_a^s) \\
&\geq \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a)(W_i^{\sigma'}(t_{g(a)}^{h(s)}) - \epsilon_{t_a^s,i}^R - \sum_{j\in\mathcal{H}_0^{k-1}} \epsilon_j^0 \overline{W}) \\
&\geq \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a) W_i(t_{g(a)}^{h(s)}) - \epsilon_{s,i}^R - \sum_{j\in\mathcal{H}_0^{k-1}} \epsilon_j^0 \overline{W}) \\
&= \sum_{a'\in A_{h(s)}} \sum_{a\in g^{-1}(a')} \sigma^{\uparrow\sigma'}(s,a) W_i(t_{g(a)}^{h(s)}) - \epsilon_{s,i}^R - \sum_{j\in\mathcal{H}_0^{k-1}} \epsilon_j^0 \overline{W} \\
&= \sum_{a'\in A_{h(s)}} \sigma'(h(s),a') W_i(t_{g(a)}^{h(s)}) - \epsilon_{s,i}^R - \sum_{j\in\mathcal{H}_0^{k-1}} \epsilon_j^0 \overline{W} \\
&= W_i^{\sigma'}(h(s)) - \epsilon_{s,i}^R - \sum_{j\in\mathcal{H}_0^{k}} \epsilon_j^0 \overline{W}
\end{aligned}$$

Where the last inequality is due to the definition of $W^{\sigma'}$ and the fact that $k \notin \mathcal{H}_0$ since we're at a node where the active player is not nature. For the case where the active player is nature, we get

$$\begin{aligned}
V_i^{\sigma^{\uparrow\sigma'}}(s) &= \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a) V_i^{\uparrow\sigma'}(t_a^s) \\
&\geq \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a)(W_i^{\sigma'}(t_{g(a)}^{h(s)}) - \epsilon_{t_a^s,i}^R - \sum_{j\in\mathcal{H}_0^{k-1}} \epsilon_j^0 \overline{W}) \\
&\geq \sum_{a\in A_s} \sigma^{\uparrow\sigma'}(s,a) W_i(t_{g(a)}^{h(s)}) - \epsilon_{s,i}^R - \sum_{j\in\mathcal{H}_0^{k-1}} \epsilon_j^0 \overline{W})
\end{aligned}$$

$$= \sum_{a' \in A_{h(s)}} \sum_{a \in g^{-1}(a')} \sigma^{\uparrow\sigma'}(s,a) W_i(t^{h(s)}_{g(a)}) - \epsilon^R_{s,i} - \sum_{j \in \mathcal{H}_0^{k-1}} \epsilon^0_j \overline{W}$$
$$\geq \sum_{a' \in A_{h(s)}} (\sigma'(h(s),a') - \epsilon^0_{s,a'}) W_i(t^{h(s)}_{g(a)}) - \epsilon^R_{s,i} - \sum_{j \in \mathcal{H}_0^{k-1}} \epsilon^0_j \overline{W}$$
$$\geq \sum_{a' \in A_{h(s)}} (\sigma'(h(s),a') W_i(t^{h(s)}_{g(a)}) - \epsilon^R_{s,i} - \sum_{j \in \mathcal{H}_0^{k}} \epsilon^0_j \overline{W}$$

Again, $V_i^{\sigma^{\uparrow\sigma'}}(s) \leq W_i^{\sigma'}(h(s)) + \epsilon^R_{s,i} + \sum_{j \in \mathcal{H}_0^k} \epsilon^0_j \overline{W}$ is similarly derived.

□

Proof of Theorem 4.2

PROOF. Assume that we are given a game M and abstract game M', and strategy profiles $\sigma', \sigma, \sigma^*$, where σ is any undivided lifted strategy profile of σ', and $\sigma^* = \sigma$ except that Player i deviates to a best response. Let σ'^{ST} be a modification of σ' such that $\sigma'^{ST}(I', a') = \sigma'(I', a')$ for all I', a' except for information sets $\hat{I}'$ such that $\sigma'_i(\hat{I}') = 0$ and $\sigma'_{-i}(\hat{I}') > 0$. For such information sets, σ'^{ST}_i is a strategy that satisfies the self-trembling property. For a game that's not two-player zero-sum this may no longer be an equilibrium. However, we only use this modified strategy profile to analyze player i's response options, and thus we do it with no loss of generality. Specifically, we prove a bound on the utility gained from playing σ^*_i instead of σ_i in terms of σ'^{ST}. In the end, we use the property that σ'^{ST} and σ' must have the same payoff to i in the abstract game to get a bound with σ'.

We show that $V_i^{\sigma^*}(r) - V_i^{\sigma}(r) \leq 2\epsilon^R_i + \sum_{j \in \mathcal{H}_i} \epsilon^D_j \overline{W} + \sum_{j \in \mathcal{H}_0} \epsilon^0_j \overline{W}$ where $r \in M$ is the root node. We do this by induction on information sets for a given player i at height k. If $\sum_{\hat{n} \in I} \sigma_{-i}(\hat{n}) = 0$, then $V_i^{\sigma^*}(I) = 0 \leq W_i^{\sigma^*}(f(I)) + \epsilon^R_I$, since all payoffs are positive.

Base case: Consider any information set I at height $k \in \mathcal{H}_i$ where k is the lowest height of information sets for i. Using Proposition 2.5, we wish to bound the following expression:

$$V_i^{\sigma^*}(I) = \sum_{a \in A_I} \sigma^*(I,a) \frac{\sum_{s \in I} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z \in Z_{t^s_a}} \sigma^*[t^s_a \to z] V_i(z) \tag{13}$$

We partition the nodes in I into the two sets $I_\phi, I_{\neg\phi}$ and bound the expression $\frac{\sum_{s \in I} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z \in Z_{t^s_a}} \sigma^*[t^s_a \to z] V_i(z)$ separately for the two cases. For I_ϕ, we apply the definition of reward approximation error to get:

$$\frac{\sum_{s \in I_\phi} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z \in Z_{t^s_a}} \sigma^*[t^s_a \to z] V_i(z)$$
$$\leq \frac{\sum_{s \in I_\phi} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z \in Z_{t^s_a}} \sigma^*[t^s_a \to z] \min_{s' \in \phi(s)} (W(\phi_{s,s'}(z)) + \epsilon^R_s)$$
$$\leq \frac{\sum_{s \in I_\phi} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z \in Z_{t^s_a}} \sigma^*[t^s_a \to z] \min_{s' \in \phi(s)} W(\phi_{s,s'}(z)) + \sigma^*_{-i}(I_\phi) \epsilon^R_s$$

The last inequality is obtained by using the fact that $\sum_{z \in Z_{t_a^s}} \sigma^*[t_a^s \to z] = 1$ because it is a probability distribution. We now use the fact that Proposition 3.2 applies to all $s' \in \phi(s)$:

$$\leq \frac{\sum_{s \in I_\phi} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \min_{s' \in \phi(s)} \sum_{z' \in Z'_{t^{s'}_{g(a)}}} (\sigma'[t^{s'}_{g(a)} \to z'] + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0) W(z') + \sigma^*_{-i}(I_\phi)\epsilon_s^R$$

$$\leq \frac{\sum_{s \in I_\phi} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \min_{s' \in \phi(s)} \sum_{z' \in Z'_{t^{s'}_{g(a)}}} \sigma'[t^{s'}_{g(a)} \to z'] W(z') + \sigma^*_{-i}(I_\phi)(\epsilon_s^R + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W})$$

Now we expand each $\sigma^*_{-i}(s)$ using the definition of distribution approximation error for nodes in I_ϕ:

$$\leq \sum_{s \in I_\phi} (\frac{\sum_{s' \in \phi(s)} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} + \epsilon_s^D) \sum_{z' \in Z'_{t^{s'}_{g(a)}}} \sigma'[t^{s'}_{g(a)} \to z'] W(z') + \sigma^*_{-i}(I_\phi)(\epsilon_s^R + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W})$$

$$\leq \sum_{s \in I_\phi} \frac{\sum_{s' \in \phi(s)} \sigma'_{-i}(s')}{\sigma'_{-i}(f(I))} \sum_{z' \in Z'_{t^{s'}_{g(a)}}} \sigma'[t^{s'}_{g(a)} \to z'] W(z') + \sigma^*_{-i}(I_\phi)(\epsilon_s^R + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W}) + \epsilon_s^D \overline{W} \tag{14}$$

For the case of $I_{\neg\phi}$, we rewrite $\sum_{s \in I_{\neg\phi}}$ as $\sum_{s' \in f(I_{\neg\phi})} \sum_{s \in h_I^{-1}(s')}$ and apply the definition of reward approximation error to get:

$$\frac{\sum_{s \in I_{\neg\phi}} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z \in Z_{t_a^s}} \sigma^*[t_a^s \to z] V_i(z)$$

$$\leq \sum_{s' \in f(I_{\neg\phi})} \frac{\sum_{s \in h_I^{-1}(s')} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z \in Z_{t_a^s}} \sigma^*[t_a^s \to z] W(h(z)) + \sigma^*_{-i}(I_{\neg\phi})\epsilon_s^R$$

We now rewrite the summation $\sum_{z \in Z_{t_a^s}}$ as $\sum_{z' \in Z'_{t^{s'}_{g(a)}}} \sum_{zo \in h^{-1}_{s'}}$, apply Proposition 3.2 and move the error term outside to get:

$$\leq \sum_{s' \in f(I_{\neg\phi})} \frac{\sum_{s \in h_I^{-1}(s')} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{z' \in Z'_{t^{s'}_{g(a)}}} \sigma'[t^{s'}_{g(a)} \to z'] W(z') + \sigma^*_{-i}(I_{\neg\phi})(\epsilon_s^R + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W})$$

Applying the definition of distribution approximation error gives:

$$\leq \sum_{s' \in f(I_{\neg\phi})} (\frac{\sigma'_{-i}(s')}{\sigma'_{-i}(f(I))} + \epsilon_{s'}^D) \sum_{z' \in Z'_{t^{s'}_{g(a)}}} \sigma'[t^{s'}_{g(a)} \to z'] W(z') + \sigma^*_{-i}(I_{\neg\phi})(\epsilon_s^R + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W})$$

$$\leq \sum_{s' \in f(I_{\neg\phi})} \frac{\sigma'_{-i}(s')}{\sigma'_{-i}(f(I))} \sum_{z' \in Z'_{t^{s'}_{g(a)}}} \sigma'[t^{s'}_{g(a)} \to z'] W(z') + \sigma^*_{-i}(I_{\neg\phi})(\epsilon_s^R + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W}) + \epsilon_{s'}^D \overline{W} \tag{15}$$

Taking the sum of equations 14 and 15, we can bound Equation 13 above by:

$$\sum_{a\in A_I} \sigma^*(I,a)\frac{\sum_{s'\in f(I)} \sigma'_{-i}(s')}{\sigma'_{-i}(f(I))} \sum_{z' Z'_{t^{s'}_{g(a)}}} \sigma'[t^{s'}_{g(a)} \to z']W(z') + \epsilon^R_s + \epsilon^D_I + \sum_{j\in\mathcal{H}^k_0} \epsilon^0_j \overline{W} \tag{16}$$

Since σ'^{ST} satisfies the self-trembling equilibrium property for i, $\sigma'^{ST}(f(I),a')$ must be at least as good as $\sum_{a\in h^{-1}(a')} \sigma^*(I,a)$ for all $a' \in A_{f(I)}$ in the abstract game. Thus, we can bound Equation 16 above by:

$$\sum_{a'\in A_{f(I)}} \sigma'^{ST}(f(I),a')\frac{\sum_{s'\in f(I)} \sigma'_{-i}(s')}{\sigma'_{-i}(f(I))} \sum_{z'\in Z_{t^{s'}_{a'}}} \sigma^{'}[t^{s'}_{a'} \to z']W_i(z') + \epsilon^R_I + \epsilon^D_k \overline{W} + \sum_{j\in\mathcal{H}^k_0} \epsilon^0_j \overline{W}$$
$$= W_i^{\sigma'^{ST}}(f(I)) + \epsilon^R_I + \epsilon^D_k \overline{W} + \sum_{j\in\mathcal{H}^k_0} \epsilon^0_j \overline{W}$$

If $\sigma'_i(f(I)) > 0$ this also holds for σ'.

Inductive step: Assume that for all I at height l, where l is the first height below k in $\mathcal{H}_i$:

$$V_i^{\sigma^*}(I) \leq W_i^{\sigma'^{ST}}(f(I)) + \epsilon^R_{I,i} + \sum_{j\in\mathcal{H}^l_i} \epsilon^D_j \overline{W} + \sum_{j\in\mathcal{H}^l_0} \epsilon^0_j \overline{W}$$

We show it holds for I at height k.

Now we use the fact that abstraction functions are partitions to write out the summations, and then apply $\sigma^*_{-i}(\hat{s}) = \sigma^*_{-i}(s)\sigma^*_{-i}[s \to \hat{s}]$ for any predecessor s. As defined in Section 2, $\mathcal{D}^a_I$ is the set of descendant information sets at the next height $l \in \mathcal{H}_i$ below k that can be reached with action a.

$$\begin{aligned}
V_i^{\sigma^*}(I) &= \sum_{a\in A_I} \sigma^*(I,a) \sum_{\hat{I}\in\mathcal{D}^a_I} \frac{\sum_{\hat{s}\in\hat{I}} \sigma^*_{-i}(\hat{s})}{\sigma^*_{-i}(I)} V_i^{\sigma^*}(\hat{I}) \\
&= \sum_{a\in A_I} \sigma^*(I,a) \sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}} \sum_{\hat{I}\in f^{-1}(\hat{I}')\cap\mathcal{D}^a_I} \frac{\sum_{s\in I}\sum_{\hat{s}\in\hat{I}\cap\mathcal{P}_s} \sigma^*_{-i}(\hat{s})}{\sigma^*_{-i}(I)} V_i^{\sigma^*}(\hat{I}) \\
&= \sum_{a\in A_I} \sigma^*(I,a) \sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}} \sum_{\hat{I}\in f^{-1}(\hat{I}')\cap\mathcal{D}^a_I} \frac{\sum_{s\in I}\sum_{\hat{s}\in\hat{I}\cap\mathcal{P}_s} \sigma^*_{-i}(s)\sigma^*_{-i}[t^s_a \to \hat{s}]}{\sigma^*_{-i}(I)} V_i^{\sigma^*}(\hat{I}) \\
&= \sum_{a\in A_I} \sigma^*(I,a) \frac{\sum_{s\in I} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}} \sum_{\hat{I}\in f^{-1}(\hat{I}')\cap\mathcal{D}^a_I} \sum_{\hat{s}\in\hat{I}\cap\mathcal{P}_s} \sigma^*_{-i}[t^s_a \to \hat{s}] V_i^{\sigma^*}(\hat{I}) \\
&\leq \sum_{a\in A_I} \sigma^*(I,a) \frac{\sum_{s\in I} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)} \sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}} \sum_{\hat{I}\in f^{-1}(\hat{I}')\cap\mathcal{D}^a_I} \sum_{\hat{s}\in\hat{I}\cap\mathcal{P}_s} \sigma^*_{-i}[t^s_a \to \hat{s}] W_i^{\sigma'}(\hat{I}') \\
&\quad + \epsilon^R_I + \sum_{j\in\mathcal{H}^l_i} \epsilon^D_j \overline{W} + \sum_{j\in\mathcal{H}^l_0} \epsilon^0_j \overline{W}
\end{aligned} \tag{17}$$

The last equality is obtained by moving $\frac{\sum_{s\in I} \sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}$ to the left. The last inequality is obtained by applying the inductive assumption and moving the error terms outside. We now bound the following expression:

$$\frac{\sum_{s\in I}\sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{I}\in f^{-1}(\hat{I}')\cap\mathcal{D}^a_I}\sum_{\hat{s}\in\hat{I}\cap\mathcal{P}_s}\sigma^*_{-i}[t^s_a\to\hat{s}]W^{\sigma'}_i(\hat{I}') \tag{18}$$

As with the base case, we bound this separately for the two sets $I_\phi, I_{\neg\phi}$. For I_ϕ, we use the two facts (1) that Proposition 3.2 applies to all $s'\in\phi(s)$, and (2) that for each descendant $\hat{s}$ of s, the abstract node $\phi_{s,s'}(\hat{s})$ must be in the same information set for all $s'\in\phi(s)$ to get:

$$\frac{\sum_{s\in I_\phi}\sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{I}\in f^{-1}(\hat{I}')\cap\mathcal{D}^a_I}\sum_{\hat{s}\in\hat{I}\cap\mathcal{P}_s}\sigma^*_{-i}[t^s_a\to\hat{s}]W^{\sigma'}_i(\hat{I}')$$

$$\leq\frac{\sum_{s\in I_\phi}\sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}\min_{s'\in\phi(s)}\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{s}'\in\hat{I}'\cap\mathcal{P}_{s'}}(\sigma'_{-i}[t^{s'}_{g(a)}\to\hat{s}]+\sum_{j\in\mathcal{H}_0,k\geq j>l}\epsilon^0_j)W^{\sigma'}_i(\hat{I}')$$

$$\leq\frac{\sum_{s\in I_\phi}\sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}\min_{s'\in\phi(s)}\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{s}'\in\hat{I}'\cap\mathcal{P}_{s'}}\sigma'_{-i}[t^{s'}_{g(a)}\to\hat{s}]W^{\sigma'}_i(\hat{I}')+\sigma^*_{-i}(I_\phi)\sum_{j\in\mathcal{H}_0,k\geq j>l}\epsilon^0_j\overline{W}$$

Now we apply the definition of distribution approximation error to get our bound:

$$\leq\sum_{s\in I_\phi}(\frac{\sum_{s'\in\phi(s)}\sigma'_{-i}(s)}{\sigma'_{-i}(I)}+\epsilon^D_s)\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{s}'\in\hat{I}'\cap\mathcal{P}_{s'}}\sigma'_{-i}[t^{s'}_{g(a)}\to\hat{s}]W^{\sigma'}_i(\hat{I}')+\sigma^*_{-i}(I_\phi)\sum_{j\in\mathcal{H}_0,k\geq j>l}\epsilon^0_j\overline{W} \tag{19}$$

For $I_{\neg\phi}$, we apply Proposition 3.2 to get:

$$\frac{\sum_{s\in I_{\neg\phi}}\sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{I}\in f^{-1}(\hat{I}')\cap\mathcal{D}^a_I}\sum_{\hat{s}\in\hat{I}\cap\mathcal{P}_s}\sigma^*_{-i}[t^s_a\to\hat{s}]W^{\sigma'}_i(\hat{I}')$$

$$\leq\frac{\sum_{s\in I_{\neg\phi}}\sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{s}'\in\hat{I}'\cap\mathcal{P}_{s'}}(\sigma'_{-i}[t^{h(s)}_{g(a)}\to\hat{s}']+\sum_{j\in\mathcal{H}_0,k\geq j>l}\epsilon^0_j)W^{\sigma'}_i(\hat{I}')$$

$$\leq\frac{\sum_{s\in I_{\neg\phi}}\sigma^*_{-i}(s)}{\sigma^*_{-i}(I)}\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{s}'\in\hat{I}'\cap\mathcal{P}_{s'}}\sigma'_{-i}[t^{h(s)}_{g(a)}\to\hat{s}']W^{\sigma'}_i(\hat{I}')+\sigma^*_{-i}(I_{\neg\phi})\sum_{j\in\mathcal{H}_0,k\geq j>l}\epsilon^0_j\overline{W}$$

We now apply the definition of distribution approximation error to get our bound:

$$\leq\sum_{s'\in f(I_{\neg\phi})}(\frac{\sigma'_{-i}(s')}{\sigma'_{-i}(f(I))}+\epsilon^D_{s'})\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{s}'\in\hat{I}'\cap\mathcal{P}_{s'}}\sigma'_{-i}[t^{s'}_{g(a)}\to\hat{s}']W^{\sigma'}_i(\hat{I}')+\sigma^*_{-i}(I_{\neg\phi})\sum_{j\in\mathcal{H}_0,k\geq j>l}\epsilon^0_j\overline{W} \tag{20}$$

Adding Equations 19 and 20 together and substituting them into Equation 17 gives us an upper bound on that equation:

$$\sum_{a\in A_I}\sigma^*(I,a)\sum_{\hat{I}'\in\mathcal{D}^{g(a)}_{f(I)}}\sum_{\hat{s}'\in\hat{I}'}\frac{\sigma'_{-i}(\hat{s}')}{\sigma'_{-i}(f(I))}W^{\sigma'}_i(\hat{I}')+\epsilon^R_I+\sum_{j\in\mathcal{H}^k_i}\epsilon^D_j\overline{W}+\sum_{j\in\mathcal{H}^k_0}\epsilon^0_j\overline{W}$$

Since σ'^{ST} is utility maximizing for Player i, we get:

$$\leq \sum_{a' \in A_{f(I)}} \sigma'^{ST}(f(I), a') \sum_{\hat{I}' \in \mathcal{D}^{\sigma'_{-i}}_{f(I),a',i}} \frac{\sum_{\hat{s}' \in \hat{I}'} \sigma'_{-i}(\hat{s}')}{\sigma'_{-i}(f(I))} W_i^{\sigma'^{ST}}(\hat{I}') + \epsilon_I^R + \sum_{j \in \mathcal{H}_i^k} \epsilon_j^D \overline{W} + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W}$$

$$= W_i^{\sigma'^{ST}}(f(I)) + \epsilon_I^R + \sum_{j \in \mathcal{H}_i^k} \epsilon_j^D \overline{W} + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W}$$

Since the only differences between σ' and σ'^{ST} are on information sets reached with probability zero, the reward from playing the two strategies must be the same. Thus we get:

$$= W_i^{\sigma'}(f(I)) + \epsilon_I^R + \sum_{j \in \mathcal{H}_i^k} \epsilon_j^D \overline{W} + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W}$$

Now, if Player i is the root player, we get that

$$V_i^{\sigma^*}(r) \leq W_i^{\sigma'}(f(r)) + \epsilon_i^R + \sum_{j \in \mathcal{H}_i^k} \epsilon_j^D \overline{W} + \sum_{j \in \mathcal{H}_0^k} \epsilon_j^0 \overline{W} \tag{21}$$

and if Player i is not the root player, we can get the same result by creating a new strategically equivalent game by adding a single node with a single action for Player i, that leads to r. Using Proposition 4.1 we get

$$V_i^{\sigma^*}(r) \leq V_i^{\sigma}(r) + 2\epsilon_i^R + \sum_{j \in \mathcal{H}_i^k} \epsilon_j^D \overline{W} + \sum_{j \in \mathcal{H}_0} 2\epsilon_j^0 \overline{W}$$

Since we assumed that σ' is an undivided lifted strategy, Proposition 3.1 gives

$$V_i^{\sigma^*}(r) \leq V_i^{\sigma}(r) + 2\epsilon_i^R + \sum_{j \in \mathcal{H}_i \cup \mathcal{H}_0} 2\epsilon_j^0 \overline{W}$$

Which is the result we wanted. □

It is possible to make the nature error bound tighter in the above, by not using the error over the entire level at a given height k, but instead keeping the errors defined in terms of the current subtree. For expositional ease we avoided this, since we later show that this error can often be kept at 0, but point out that it could be relevant in certain applications.

Using exactly the same technique as above, except without using the self-trembling property for subgame-inducing nodes, it is possible to show that any subgame-perfect equilibrium in the abstract game maps to an approximate subgame-perfect equilibrium in the full game.

Proof of Proposition 4.3

PROOF. Consider the zero-sum game in Figure 4. In the middle of the figure we have the two possible abstractions achievable by performing a single action abstraction. On the right is the resulting abstraction that they both strategically reduce to. Clearly, the only Nash equilibrium in this game is the undivided strategy of Player 1 playing LR and Player 2 playing lr. For any undivided lifted strategy, the value is either 0 or 10. If it's 0, Player 1 can switch strategy and get 10. If it's 10, Player 2 can switch strategy and lower the value to 0. Thus, one of the two players will have regret $2\epsilon^R$. □

Proof of Proposition 4.4

Consider the game shown in Figure 5. It only has one player, but we can easily insert subgames at the leaf nodes such that these values are obtained in the subgames. The only possible abstraction that does not immediately yield full loss is to merge the two branches extending from the root node.

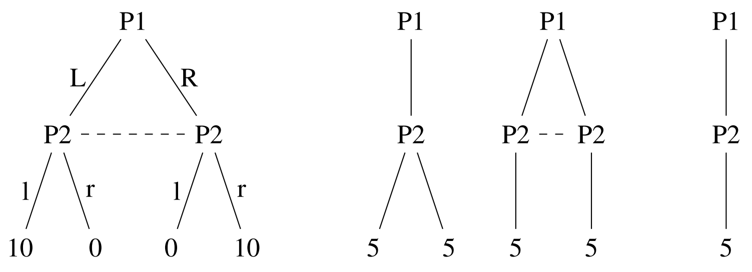


Fig. 4. On the left is a zero-sum extensive form game with one decision node for Player 1 and two decision nodes (in the same information set) for Player 2. In the middle are the two possible results from performing a single action abstraction, either merging actions L and R, or l and r. On the right is the final abstraction which both intermediate abstractions reduce to.

Doing this leads to the game shown on the right. Since any action is equally good, one equilibrium is for the player to pick the right action. In the full game, at the information set on the right, choosing the left action instead yields an extra utility of 2 at the information set. Multiplying this by the probability of reaching the set, we get a loss of 1. The theoretical bound is 6, and this gives us tightness of the bound up to a factor of 6.

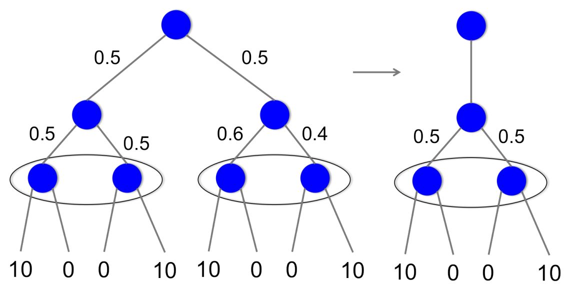


Fig. 5. On the left is a simplified extensive form game with two layers of nature nodes, followed by decision nodes for Player 1 in two different information sets. On the right is given the only possible abstraction that does not lead to a loss of 10.

Proof of Theorem 5.3

PROOF. First we show a reduction to GI. For TI, consider a given extensive form game M. We modify the game tree such that it forms a graph G_M in the following way. For each information set, we make the nodes in the information set a clique. For nature nodes, we leave them the way they are, and ignore the probability distributions. For leaf nodes, since we are only concerned with topological isomorphism, we just convert them to an unlabeled node with no value. Finally, a unique clique size is chosen for the roots to be converted to. An example is given in Figure 6. Here we have a single information set containing all the nodes at heigh 2. We turn those nodes into a clique in the transformation, and convert the utilities at leaf nodes to reagular leaf nodes. First we show that any GI mapping h computed on $G_M, G_{M'}$ corresponds to a TI on M, M'. We will do this by induction on the height k.

Consider leaf nodes at height $k = 1$. Leaf nodes are the only nodes with degree 1, and thus for any leaf node $z \in Z_M$, $h(z)$ must be a leaf node in M', and vice versa. There are no information sets to respect at the leaf level.

Now consider height k, and assume that all nodes at heights below k are correctly mapped. For any node s at height k, we have that $h(s)$ must also be at height k in M', and they must have the same

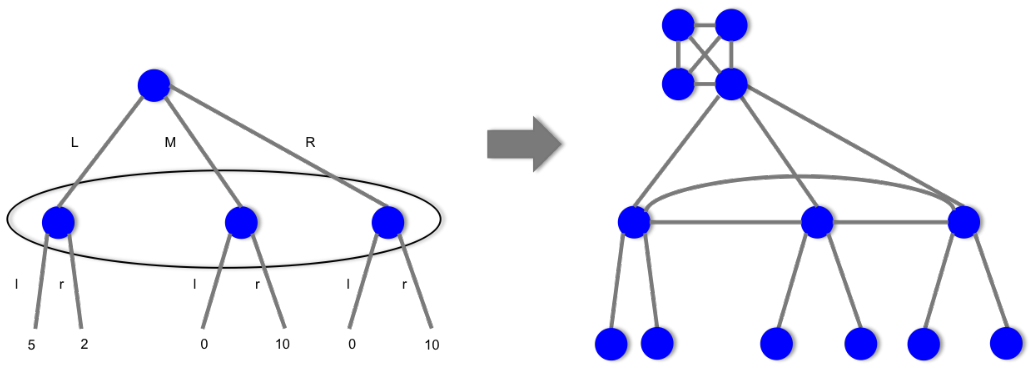


Fig. 6. On the left is an extensive form game. On the right is the resulting graph after turning the information set in to a clique, and converting utilities at leaf nodes to regular leaf nodes.

length path to $r, h(r)$ respectively, since tree isomorphism preserves levels, as described above when we introduced our isomorphism definitions. Our added edges do not change this, since they only connect vertices within the level. Now consider some information set I at height k, and $s_1, s_2 \in I$. s_1, s_2 are connected by information set edges, and thus $h(s_1), h(s_2)$ must also be connected, which is only the case if they're in an information set together. Conversely, if $s_1 \in I_1, s_2 \in I_2, I_1 \neq I_2$, then they do not have an edge between them, and neither does $h(s_1), h(s_2)$, which is only the case if $h(s_1), h(s_2)$ are in separate information sets.

Now, we show that if h is an isomorphic mapping for M, M' then it also defines an isomorphic mapping for $G_M, G_{M'}$. By the same reasoning as above, we have that all nodes must be at their respective levels, and leaf nodes must be correct. For $s_1, s_2 \in I$ at some level k, we have that $h(s_1), h(s_2)$ share an information set as well, and thus the nodes are connected by an edge in both augmented graphs. Similarly, for $s_1 \in I_1, s_2 \in I_2, I_1 \neq I_2$, $h(s_1), h(s_2)$ do not share an information set, and thus the nodes are not connected by an edge in either of the augmented graphs.

For SI, we perform the same transformation as above, except for leaf nodes and nature nodes. For every nature node type, we insert a separate unique structure. Then only nature nodes of the same type can map to each other. Further, for nature nodes that have uniform distribution over their actions, we can treat them as normal nodes, since they will only be merging with other nature nodes at the same level with uniform distribution, and the same number of actions. This can be any unique structure, one option is to use cliques of varying sizes, one size for each type, where the path from the root enters by some node in the clique, and leaves from some other node. This is a separate unqiue structure from the information set cliques, since those had edges going in and out of the clique from every node in the clique. Further, to make sure that we map the correct probability edges to each other, we can insert another unique structure along each unique edge probability, for example a circle, with circles of length $\{3, \dots, 3 + |\text{nature edge types}|\}$.

Now consider some leaf node $z \in Z_M$, with values $v_1, \dots, v_n$ for the players. From the node z, we insert a clique of size $i + 2$ for each player i, and add an edge from some node u in the clique to z. We then construct a cycle graph of size $c(v_i)$, and connect u to some node in the cycle graph. $c(v_i)$ is the smallest unique cycle size. This type of clique is again unique, since edges enter and leave the clique from the same node u, which is different from the nature cliques. Now, for two leaf nodes $z \in M, z' \in M'$, we have that they are only isomorphic to each other if they have the same payoffs for each player, since each player has a uniquely sized clique, and the circles at the cliques must be of the same size. An example is given in Figure 7. On the left is a leaf node with payoffs $[2, 2, 3]$ for players $1, 2, 3$ respectively. On the right is the resulting subgraph that replaces the leaf node. At the top is the leaf node. Connected to the leaf are cliques, one for each player, of size $2 + i$

for Player i. For each clique, we have the value that the player attains as a cycle graph connected to that player's clique.

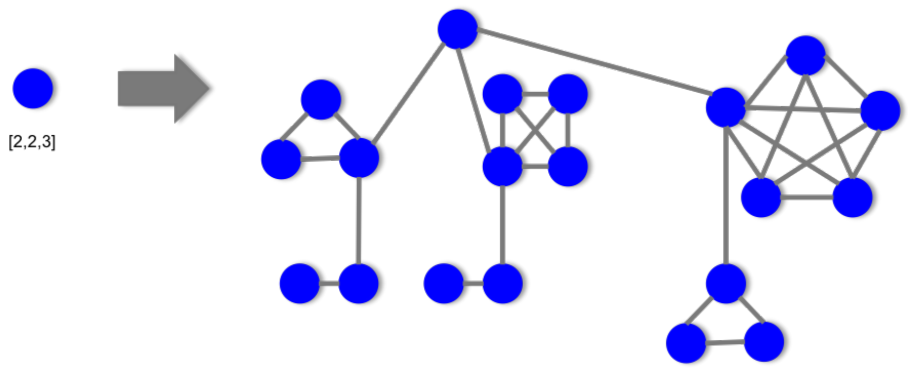


Fig. 7. On the left is a leaf node with payoffs $[2, 2, 3]$ for players $1, 2, 3$ respectively. On the right is the subgraph that replaces the lead node in the reduction.

Now we show that GI reduces to TI, which suffices to show GI completeness, since TI is a special case of SI. Given a graph $G = \{V, E\}$, we construct the following extensive form game. First we add a root node. For each vertex $v \in V$, we add a node s_v descending from the root node, in an information set by itself. Thus, at depth two, we have $|V|$ nodes. Now, for any two nodes $v, \hat{h} \in V$ that share an edge, we create nodes $s_{v,\hat{v}}, s_{\hat{v},v}$ at depth three, each descending from their respective nodes $s_v, s_{\hat{v}}$, and put them in an information set together.

Now consider determining isomorphism between graphs $G = \{V, E\}, G' = \{V', E'\}$ given an isomorphic mapping h between their game representation. If nodes $s_v, s_{\hat{v}}$ share an information set at their child nodes, then $h(s_v), h(s_{\hat{v}})$ must share an information set at their child nodes. However, this can only happen if both sets share an edge. Conversely, if nodes $s_v, s_{\hat{v}}$ do not share an information set at their child nodes, then $h(s_v), h(s_{\hat{v}})$ also do not share an information set at their child nodes. This implies that neither of the sets share an edge in their respective graphs. Thus, we get that any game isomorphism h defines a graph isomorphism as well. We can apply the exact same logic to achieve the converse result.

We have shown that both TI and SI can be reduced to GI, and that GI reduces to TI. Thus, we have that TI and SI are GI-complete. $\square$

Proof of Theorem 5.5

PROOF. The problem is easily seen to be in NP. To show NP-hardness, we reduce from SAT. Consider some SAT problem consisting of variables V and clauses C. For each variable $v \in V$, we construct an action for each of the literals $v, \neg v$. We let the cost of merging these actions be 0, and the cost of merging two literals that are not from the same variable be M. Here M is some large number. Now, for each clause, we make an action. We make the cost of merging this action with any other clause or any literal not in the clause M, and 0 if merged with a literal in the clause. We also add add an extra dummy action for each variable, which has cost 0 of merging with either of the two literals in the clause, and M for everything else. Now we ask if there is some abstraction with $|V|$ actions and cost zero. Clearly, the clauses cannot be chosen as prototypes without incurring a loss of M, because of the dummy actions. Instead, for each variable one of the two literals must be chosen as a prototype, and if there is such a set, it means that all clauses can map onto some literal, and must therefore be satisfied. Conversely, if there is a satisfying assignment, then we can pick the literals in the assignment as prototypes, and the clauses can be mapped onto the chosen literals with 0 loss. $\square$

Abstractions generated by the IP in the experiments

In this section, we show the four lossy abstractions computed by our IP model. The lossless abstraction that it computes is too large to show here, but is the same as the canonical representation of poker hands, where all suit isomorphisms are removed.

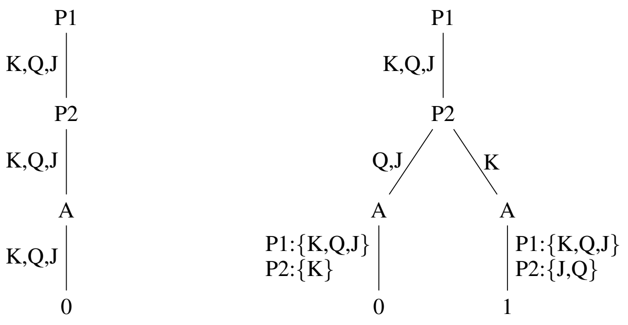


Fig. 8. Left: The coarsest abstraction that our algorithm found. That abstraction treats all cards as the same, so players essentially play blind. Right: The second-coarsest abstraction that our algorithm found. Interestingly, the card abstraction is not the same for the two players.

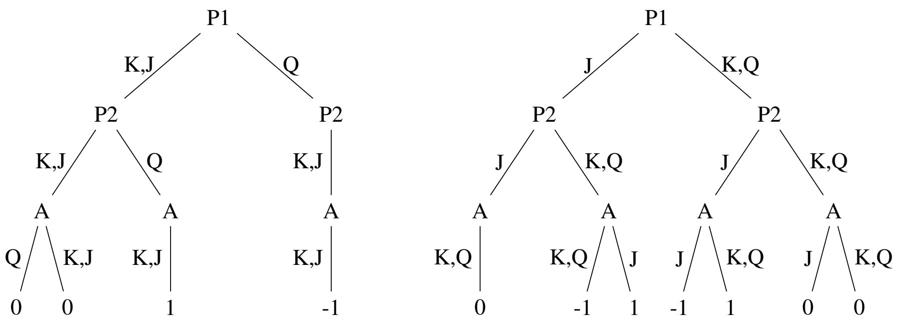


Fig. 9. Left: The third-coarsest abstraction that our IP found. Right: The fourth-coarsest (and finest lossy) abstraction that our IP found.

Experiment

We applied our IP model to a simple poker game. One advantage of poker games for testing our approach is that the chance outcomes can be decoupled from the player actions using the signal tree. Any abstraction computed on the signal tree can easily be converted to an abstraction of the full game. Since we are solving an NP-complete problem, it is desirable that we can apply it to a smaller structure, such as the signal tree, while having a fairly complex full game for investigating exploitability.

Our game has a deck of five cards: two jacks, a queen, and two kings. Two players play the game, and each antes one chip. Each player is then dealt a private card from the deck. This is followed by a betting round. If neither player folded, one public card is dealt face up, and another betting round occurs. After the second betting round, if neither player has folded, a showdown occurs; the player with the strongest hand wins. If a player pairs his private card with the public card he wins the showdown. Otherwise, the player with the highest card wins it.

Player 1 starts each betting round. He can check or bet. If he checks, Player 2 can check or bet. Whenever a player has bet, the opponent can call, fold, or raise. The round ends when a player calls or both players check. Only one raise (beyond a bet) is allowed per round. In the first betting round, bets and raises are two chips, and in the second betting round they are four chips.

We briefly described the signal tree for a poker game in Section 5.1.3. Gilpin and Sandholm [2007b] introduced it in the specific setting of games of ordered signals, a game class closely resembling poker. More generally, in any game where the player's action options are independent of nature's moves, abstraction can be performed on the signal tree. The signal tree is defined conceptually by removing all player actions from the game tree, and only considering the tree of possible nature moves. Clearly, for this to be possible, the nature moves can be conditioned only on which prior nature events have occurred, not on player actions.

Since information sets related to nature's moves are entirely decided by the signal tree, it is possible to apply our IP model directly to the signal tree, and use the abstraction generated on the signal tree to construct an abstraction of the full game. The unabstracted signal tree has 86 nodes; the game tree has 4806 nodes. The IP model has approximately 4,500 binary variables and 40,000 constraints. To solve the model, we used CPLEX version 12.5.

For the model where a bound is given as input and the objective is to minimize tree size, we ran experiments with a bound ranging from 0 to 20 chips. Figure 3 Left shows a plot of the game sizes as a function of the bound. As can be seen, tree size is a step function of the given bound. Four different abstraction sizes were found. The coarsest abstraction has four signal tree nodes, and thus represents a single sequence of outcomes where the players act essentially without seeing any cards. The coarsest lossless abstraction has size 30. It is not until we allow a loss bound of 5 that the algorithm finds a lossy abstraction (of size 14).

For the model where a maximum tree size is given as input and the objective is to minimize the, we ran experiments for signal tree size bounds from 4 to 54. Figure 3 Right shows exploitability as a function allowed signal tree size. Three plots are shown, the exploitability of each of the two players, and the exploitability bound given by Theorem 4.2. By and large, the three plots decrease with signal tree size. However, we see a non-monotonicity at size 6, where Player 1's exploitability goes up compared to that at size 4. This is an instance of abstraction pathology, which exists in games: refining an abstraction game cause the equilibrium strategies to have higher exploitability in the original game [Waugh et al. 2009a]. Finally, when we allow a tree size of 30, the lossless abstraction is found, and beyond that all experiments show zero exploitability.

The different abstractions (signal trees) that the IP generated are presented in the appendix. Interestingly, sometimes the IP generated an abstraction that is asymmetric between the players.