# Simplifying Imperfect Recall Games

Hugo Gimbert
LaBRI, CNRS, Université de Bordeaux
Talence, France
hugo.gimbert@cnrs.fr

Soumyajit Paul
University of Liverpool
Liverpool, United Kingdom
soumyajit.paul@liverpool.ac.uk

B. Srivathsan
Chennai Mathematical Institute and
CNRS, ReLaX, IRL 2000, Siruseri, India
sri@cmi.ac.in

## ABSTRACT

In games with imperfect recall, players may forget the sequence of decisions they made in the past. When players also forget whether they have already encountered their current decision point, they are said to be absent-minded. Solving one-player imperfect recall games is known to be NP-hard, even when the players are not absent-minded. This motivates the search for polynomial-time solvable subclasses. A special type of imperfect recall, called *A-loss recall*, is amenable to efficient polynomial-time algorithms. In this work, we present novel techniques to simplify non-absent-minded imperfect recall games into equivalent A-loss recall games. The first idea involves shuffling the order of actions, and leads to a new polynomial-time solvable class of imperfect recall games that extends A-loss recall. The second idea generalises the first one, by constructing a new set of action sequences which can be "linearly combined" to give the original game. The equivalent game has a simplified information structure, but it could be exponentially bigger in size (in accordance with the NP-hardness). We present an algorithm to generate an equivalent A-loss recall game with the smallest size.

## KEYWORDS

Games; Imperfect information games; Imperfect recall

**ACM Reference Format:**
Hugo Gimbert, Soumyajit Paul, and B. Srivathsan. 2025. Simplifying Imperfect Recall Games. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

## 1 INTRODUCTION

Games play a central role in AI research. In the early $20^{th}$ century, [25] showed that perfect information games in extensive form can be solved by a bottom-up traversal of the game tree. Despite the fact that this does not readily provide efficient ways to solve large games such as Chess or Go in practice, this has indeed laid the foundation for the dramatic progress in the field of perfect information games, with computer programs being able to challenge human experts. Solving games becomes more intricate when the players (agents) have incomplete information about the state of the game – Poker for instance, where a player does not know the cards of the others. One of the remarkable imperfect information games where computer programs have been able to defeat professional human players is Texas Hold'em Poker [2, 3, 18]. A main technique used in these algorithms is the abstraction of large games into smaller *imperfect recall* games.

Perfect recall is the ability of a player to remember her own actions. Poker is an imperfect information game played by several players. However, ideally one would assume that the players have a perfect recall of their actions. An imperfect recall player does not remember the sequence of her own actions. Imperfect recall allows for a structured mechanism to forget the information history and as [21] argues, it is particularly suited for AI agents.

From a modeling perspective, imperfect recall has been used to describe teams of agents, where each team can be represented as a single agent with imperfect recall [4, 22] or to describe agents modeling multiple nodes which do not share information between each other due to privacy reasons [7]. Moreover [16] argues that imperfect recall is a model of bounded rationality. Given the limited memory of players, it is not realistic to assume that the players remember all their actions. We refer the reader to [21] for an excellent introduction to different uses of imperfect recall. From a practical perspective, the most prominent use of imperfect recall is in abstracting games [1, 8, 23, 26]. The state space generated by usual games is typically very large and abstractions are crucial for solving such games. Abstractions that preserve perfect recall force a player to distinguish the current information gained, in all later rounds, even if it is not relevant. Abstractions using players with imperfect recall have been shown to outperform those using players with perfect recall [1, 6, 11, 24].

From a complexity perspective, imperfect recall games are known to be NP-hard [5, 14] even when there is a single player, whereas perfect recall games can be solved in polynomial-time [14, 19]. Recent studies have aligned the complexity of different solution concepts for imperfect recall games to the modern complexity classes [9, 20, 21]. The hardness of imperfect recall games has motivated the search for subclasses which are polynomial-time solvable [12, 13], or where algorithms similar to the perfect recall case can be applied [15, 17]. The class of *A-loss recall* [12, 13] is a special kind of imperfect recall, where the loss of information can be traced back to a player forgetting her own action at a point in the past – the player remembers *where* it was played, but forgets *what* was played. We consider A-loss recall games to be *simple* since there are polynomial-time algorithms for solving them. To the best of our knowledge, A-loss recall games are the biggest known class of imperfect recall with a polynomial-time solution. This has led to research towards finding A-loss recall abstractions [5].

*Contributions.* Our broad goal in this work is to find efficient ways to solve imperfect recall games in extensive-form. We do so by simplifying them into A-loss recall games. We focus on games where the players are not absent-minded: a player is absent-minded if she even forgets whether a decision point was previously seen or not. Here are our major contributions.



*Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA.* © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

(1) We first identify a class of one-player games where the player's information structure is more complex than A-loss recall, but shuffling the order of actions results in an equivalent A-loss recall game. This leads to a new PTIME solvable class of imperfect recall games, that extends A-loss recall (Theorem 2, Corollary 2, Corollary 3). Furthermore, these classes themselves can be tested in PTIME.
(2) We show that every game with *non-absentminded* players can be transformed into an equivalent A-loss recall game (Theorem 3). We present an algorithm to generate an equivalent A-loss recall game with the smallest size.

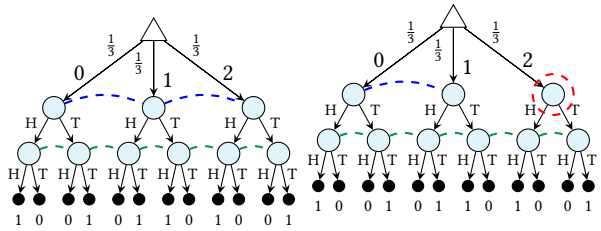
The caveat in the second result above is that the resulting A-loss recall game could be exponentially bigger. This is expected, since solving imperfect recall games is NP-hard, whereas A-loss recall games can be solved in PTIME. The result however shows that in order to solve imperfect recall games, one could either use a worst-case exponential-time algorithm on the original game, or apply our transformation to a worst-case exponential-sized game and run a polynomial-time algorithm on it. From a conceptual point of view, our result shows that as long as there is no absentmindedness, imperfect recall can be transformed into one where the information loss can be attributed to forgetting own actions at a past point.

*Organization of the document.* Section 2 introduces a modification of the popular matching pennies game that will be used as a running example to illustrate our results. Section 3 recalls necessary preliminaries on extensive-form games. Section 4 presents the new polynomial-time class of shuffled A-loss recall. Section 5 generalizes the idea of shuffling to incorporate a "linear combination" of action sequences, and presents the second result mentioned above. Section 6 extends the results to the two-player setting. Missing proofs and details can be found in the extended version[10].

## 2 AN EXAMPLE

Let us start with a one-player game called the *single team matching-unmatching pennies game*, which will be used as a running example. A team of players with the same goal can be interpreted as a single player. In this case, the team consists of two players Alice and Bob, each possessing a coin with two sides, Head (H) and Tail (T) and each of them must choose a side for their respective coins independently. The game unfolds in the following manner : a fair $n$-faced die with outcomes from $\{0, \ldots, n-1\}$ is rolled; then Alice chooses a side from $\{H, T\}$, followed by Bob choosing from $\{H, T\}$. Winning or losing depends on the parity of the die outcome. If the outcome of the die is even, then they win if and only if they match their sides. If the outcome is odd, they win if and only if their sides do not match. We consider three variants depending on what Alice and Bob can observe, and model it in extensive form in Fig. 1 for $n = 3$. An informal description of the figures follows after this paragraph.

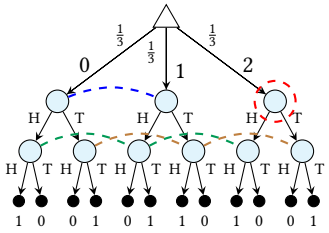
**I.** Both Alice and Bob observe nothing (Fig. 1a).
**II.** Alice can't distinguish between die outcome $2i$ and $2i+1$ for $i \geq 0$, but Bob observes nothing (Fig. 1b).
**III.** Alice can't distinguish between die outcome $2i$ and $2i+1$ for $i \geq 0$, Bob only observes coin of Alice but not outcome of die (Fig. 1c).



**(a) Alice and Bob, both observe nothing**



**(b) Alice can't distinguish between $2i$ and $2i+1$ for $i \geq 0$, Bob observes nothing**



**(c) Bob only observes Alice's coin**

**Figure 1: Three versions of the single team matching-unmatching pennies game for $n = 3$**

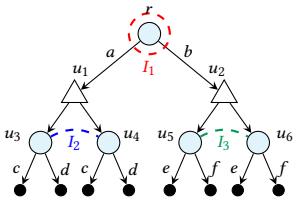
Alice and Bob want to maximize their *expected payoff*. We will see their possible strategies in Section 3. Later, we will see that game **I** falls under the simple class of A-loss recall. In Section 4 and Section 5 we will see how to simplify games **II** and **III** respectively.

Before we delve into the background and results, here is a description of the extensive-form model. The root node (the triangle), is the event of rolling the die. The triangle nodes are called Chance nodes, and the edges out of them associate probabilities to each of the outcomes. For this game, the distribution is uniform. The circle nodes denote decision nodes of the team. The nodes in the second level (root being the first) belong to Alice whereas the nodes in the third level belong to Bob. The actions labelled in edges out of these nodes denote the actions available to the corresponding players. A leaf node indicates an end state, and a path from root to leaf denotes a play from start to end. Each leaf is associated with a *payoff* that the team receives at the end of the corresponding play. E.g., in Fig. 1a in the play resulting from the path $0, H, T$ the payoff is 0 because the team loses. It is 1 when they win.

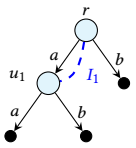
Imperfect information is expressed using a dotted line: a player cannot distinguish between two nodes joined by a dotted line. For e.g., in Fig. 1a the dotted red line joining all of Alice's nodes indicates that Alice cannot observe the die outcome. Similarly, the blue dotted line for Bob indicates, he neither observes the outcome of the die, nor the side of the coin chosen by Alice. These sets of indistinguishable nodes are called *information sets*.

## 3 BACKGROUND AND NOTATIONS

This section presents the formal definitions. The single team matching-unmatching pennies game has only one player and chance nodes, but in general we will talk about zero-sum two player games. As




**(a)** Max **with perfect recall**




**(b)** Max **with absentmindedness**

**Figure 2: Recalls of** Max

in Fig. 5, there are two players Max (circle nodes) and Min (square nodes). The payoff at the leaf, is the amount Min loses and Max gains. The goal of Max is to maximize the expected payoff whereas Min wishes to minimize it. In Fig. 1 Max was the team consisting of Alice and Bob.

In this paper, we mainly work with *game-structures* and not games themselves. Game-structures are essentially games sans the numerical quantities. Any game on a game structure can be represented symbolically as in shown Fig. 4a with symbolic payoffs $z_i$s and symbolic chance probabilities $p_i$s (with constraints on $p_i$'s). An extensive form game can be obtained from a game structure by plugging in values for $z_i$s and $p_i$s. We work with game structures because the notions of perfect recall and imperfect recall can be determined simply by looking at the game-structure.

Formally, a game-structure $\mathcal{T}$ is a tuple $(V, L, r, A, E, \mathcal{I})$ where $V$ is a finite set of non-terminal nodes partitioned as $V_{\mathsf{Max}}$, $V_{\mathsf{Min}}$ and $V_{\mathsf{Chance}}$; $L$ is a finite set of leaves; $r \in V$ is a root node; $A = A_{\mathsf{Max}} \cup A_{\mathsf{Min}}$ is a finite set of actions; $E \subseteq V \times (V \cup L)$ is an edge relation that induces a directed tree; edges originating from $V_{\mathsf{Max}} \cup V_{\mathsf{Min}}$ are labelled with actions from $A$; we write $u \xrightarrow{a} v$ if $(u, v)$ is labelled with $a$, and assume that there is no incoming edge $u \longrightarrow r$ to the root node $r$; $\mathcal{I} = \mathcal{I}_{\mathsf{Max}} \cup \mathcal{I}_{\mathsf{Min}}$ is a set of information sets for $i \in \{\mathsf{Max}, \mathsf{Min}\}$, each information set $I \in \mathcal{I}_i$ is a subset of vertices belonging to $i$, i.e. $I \subseteq V_i$, and moreover, the set of information sets $\mathcal{I}_i$ partitions $V_i$. E.g., in Fig. 2a, $\mathcal{I}_{\mathsf{Max}} = \{I_1, I_2, I_3\}$ and $I_1 = \{r\}, I_2 = \{u_3, u_4\}$ and $I_3 = \{u_5, u_6\}$. We can understand these information sets as a signal that the player receives when she reaches a node in it. On receiving the signal, the player knows the actions that are available to play at that position.

An information set models the fact that a player cannot distinguish between the nodes within it. Therefore, the set of outgoing actions from each node in an information set is required to be the same. This allows us to define $\mathrm{Act}(I)$ as the set of actions available at information set $I$. E.g., in Fig. 2a, $\mathrm{Act}(I_2) = \{c, d\}$. For technical convenience, we make a second assumption: for all $I, I' \in \mathcal{I}$ with $I \neq I'$, we have $\mathrm{Act}(I) \cap \mathrm{Act}(I') = \emptyset$. Therefore, the actions identify the information sets. With this assumption, in Fig. 1, the actions of Alice should be seen as $H_A, T_A$ and those of Bob's as $H_B, T_B$. But we omit the subscripts in the figure for clarity.

**Definition 1** (Extensive form games). *A two-player zero-sum game in extensive form is a tuple* $(\mathcal{T}, \delta, \mathcal{U})$ *where* $\mathcal{T}$ *is a game-structure,* $\delta$ *is the* chance probability *associating to each* Chance *node, a probability distribution on the outgoing actions, and* $\mathcal{U} : L \mapsto \mathbb{Q}$ *is the utility function associating a payoff to each leaf.*

The *size* of a game is the sum of the bit-lengths of all chance probabilities and leaf payoffs in it. A *behavioral strategy* for player Max (Min resp.) assigns a probability distribution to $\mathrm{Act}(I)$ for each $I \in \mathcal{I}_{\mathsf{Max}}$ ($\mathcal{I}_{\mathsf{Min}}$ resp.). Once we fix behavioral strategies $\sigma$ and $\tau$ for Max and Min respectively, each edge in the game has an associated probability of being taken, given by the corresponding strategy or Chance. The probability of reaching a leaf $u \in L$ is given by the product of all the numbers along the path to the leaf. Consider Fig. 3a. Let $\sigma$ assign $\frac{1}{4}$ to $b$ and $\frac{3}{4}$ to $\bar{b}$; 0 and 1 to $c$ and $\bar{c}$, and $\frac{1}{3}$ to $a$ and $\frac{2}{3}$ to $\bar{a}$. The probability of reaching the leaf $b\bar{a}$ is then: $p_1 \times \frac{1}{4} \times \frac{2}{3}$. For a leaf $u$, we denote this quantity by $\mathrm{Prob}_{\sigma,\tau}(u)$. The *expected payoff* $\mathbb{E}(\sigma, \tau)$ when Max plays $\sigma$ and Min plays $\tau$, then equals $\sum_{u \in L} \mathrm{Prob}_{\sigma,\tau}(u)\mathcal{U}(u)$. The solution concept that we will consider in this paper is the notion of maxmin. The *maxmin value* of a game is given by $\max_{\sigma} \min_{\tau} \mathbb{E}(\sigma, \tau)$ where $\sigma, \tau$ are behavioral strategies of Max and Min respectively. A strategy of Max which provides the maxmin value is called a *maxmin strategy*. In one-player games, we only have Max player and the maxmin value of the game is $\max_{\sigma} \mathbb{E}(\sigma)$. For one-player non-absentminded games, the maxmin value can be in fact obtained by a *pure strategy* – pure strategies are special cases of behavioural strategies which assign either 0 or 1 to each action [14].

The maxmin value of the game in Fig. 1a is $\frac{2}{3}$ since Alice and Bob can win at most in 2 of the 3 die rolls by playing matching sides. Another way to see this is to consider the four possible pure strategies $HH, HT, TH, TT$, which induce payoffs $\frac{2}{3}$, $\frac{1}{3}$, $\frac{1}{3}$ and $\frac{2}{3}$ respectively. Now since, in the rest of the following two versions, the team has more information [1] they can guarantee at least $\frac{2}{3}$ by playing the same strategy. Interestingly, one can observe (by enumerating all pure strategies) that they cannot do better than that in any version.

*Histories and recalls.* We now move on to describing the various types of imperfect information, based on what the player remembers about her history. A node $w \in V$ is reached by a unique path from the root: $r = v_0 \longrightarrow v_1 \longrightarrow \cdots \longrightarrow v_n = w$. Let $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ be the vertices in this sequence which do not belong to Chance. Then, $hist(w) = a_1 a_2 \cdots a_{k-1}$, where $v_{i_j} \xrightarrow{a_j} v_{i_{j+1}}$. For a player $i \in \{\mathsf{Max}, \mathsf{Min}\}$ the history of $i$ at $w$, denoted by $hist_i(w)$, is the sequence of player $i$'s actions in the path to $w$, which is simply the sub-sequence of $hist(w)$ restricted to actions from $A_i$. E.g.: in Fig. 2a, $hist_{\mathsf{Max}}(u_3) = hist_{\mathsf{Max}}(u_4) = a$; in Fig. 3a, $hist_{\mathsf{Max}}(u_3) = b$ and $hist_{\mathsf{Max}}(u_2) = \epsilon$, the empty sequence. It is important to remark that this definition uses the assumption that actions determine information sets – otherwise, we would need to incorporate the information sets that were visited along the way, into the history.

Let $\mathcal{H}$ denote the set of all histories and $\mathcal{H}_i$ be the set of all histories of player $i$. For an information set $I \in \mathcal{I}_i$ let $\mathcal{H}(I) = \{hist(u) \mid u \in I\}$ be the set of histories of all nodes in $I$. Similarly, we can define $\mathcal{H}_i(I)$ with respect to $\mathcal{H}_i$. Let $\mathcal{H}(L)$ denote the set of all leaf histories.

When $\mathcal{H}_i(I)$ has multiple histories, at a node $v \in I$ the player does not remember which history she traversed to reach $v$. Hence

[1] This can be observed by the fact that information sets in each version are refinements of the previous versions.

the player loses information. For two nodes $u$ and $v$ in $I$, comparing $hist_i(u)$ and $hist_i(v)$ reveals the loss or retention of previously withheld information at the respective nodes. To capture this there are different notions of *recall*.

*Perfect recall.* Player $i$ is said to have *perfect recall* (PFR) if for every $I \in \mathcal{I}_i$, and every pair of distinct vertices $u, v \in I$, we have $hist_i(u) = hist_i(v)$, i.e. $|\mathcal{H}_i(I)| = 1$. Otherwise, the player is said to have imperfect recall. Fig. 2a is an example of a perfect recall game. *Imperfect recall.* Fig. 3a gives an example of a game-structure that has imperfect recall. Notice that states $u_3$ and $u_4$ lie in the same information set $I_3$, but the sequence of the player's actions leading to these states is different: history at $u_3$ is $b$, whereas at $u_4$ it is $\bar{b}$. Within imperfect recall, there are distinctions. The imperfect recall in Fig. 3b and the one in Fig. 3a are in some sense different: in Fig. 3b, the inability to distinguish between the two nodes in $I_1$ can be traced back to a point in the past where she forgets her own action from some information set ($I_3$ in this case), whereas in Fig. 3a, the player has been able to distinguish between the two outcomes of the Chance node, but later forgets at $I_3$ where she started from, leading to four histories $b, \bar{b}, c$ and $\bar{c}$ at $I_3$.

*A-loss recall.* Game-structures as in Fig. 3b are said to have *A-loss recall.* A consequence of having A-loss recall is that a player always remembers any new information gained from Chance outcomes, which is not the case in Fig. 3a. Player $i$ has *A-loss recall* (ALR) if for all $I \in \mathcal{I}_i$, and every pair of distinct vertices $u, v \in I$, either $hist_i(u) = hist_i(v)$, or $hist_i(u)$ is of the form $sas_1$, and $hist_i(v)$ of the form $sbs_2$, where $a, b \in \text{Act}(I')$ for some $I' \in \mathcal{I}$, with $a \neq b$. The game in Fig. 1a has A-loss recall, whereas the others, Fig. 1b and Fig. 1c do not.

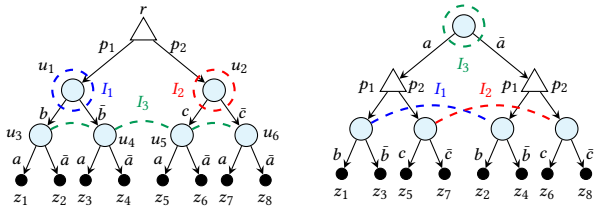
Finally, player $i$ is said to be *non-absentminded* (NAM) if $\forall u, v \in V_i$ with $u$ lying on the path to $v$, the information set that $u$ belongs to is different from the information set that $v$ belongs to, i.e. all nodes of $i$ on a path from $r$ to leaf node lie in distinct information sets. Fig. 2b is an example where Max is absentminded, since both $r$ and $u_1$ lie in the same information set. Notice that PFR impliesALR, which in turn implies implies NAM.

When Max and Min have recalls $R_{\text{Max}}, R_{\text{Min}} \in \{\text{PFR, ALR, NAM}\}$ respectively we will denote the game as a $(R_{\text{Max}}, R_{\text{Min}})$-game. A one-player game with recall $R$ is denoted as $R$-game. In this paper we are only concerned with one-player NAM-games and two-player (NAM, NAM)-games. Let us now recall some known results.

- A maxmin solution in a (PFR, ALR)-game can be computed in polynomial- time [12, 14, 19]. As a corollary, an optimal solution in a one-player ALR-game can be computed in polynomial-time [12].
- The maxmin decision problem for (NAM, NAM)-games is both NP-hard [14] and Square-Root-Sum-hard [9] [2]. The NP-hardness and the Square-Root-Sum-hardness hold even for (ALR, PFR)-games [5, 9]. The maxmin decision problem for one-player NAM-games is NP-complete [14].

Our core idea is to view game structures through the polynomials they generate.

*Leaf monomials.* In a game structure, assigning variable $x_a$ to each action $a$, the monomial obtained by taking the product of all $x_a$ along the path to each leaf $t$ is called a *leaf monomial*, and denoted as $\mu(t)$. E.g., the leaf monomials of the game-structure in Fig. 2a are $\{x_a x_c, x_a x_d, x_b x_e, x_b x_f\}$. For a game structure $\mathcal{T}$, we will write $X(\mathcal{T})$ for the set of leaf monomials. For a game $G$, let $\text{Prob}_{\text{Chance}}(t)$ denote the product of Chance probabilities in the path to $t$. The polynomial given by $\sum_{t \in L} \text{Prob}_{\text{Chance}}(t) \cdot \mathcal{U}(t) \cdot \mu(t)$ is called the *payoff polynomial* of a game. A constraint of the form $\sum_{a \in \text{Act}(I)} x_a = 1$ for an information set $I$ will be called a *strategy constraint.* Any non-negative valuation satisfying these constraints gives a behavioral strategy to the players. The maxmin value in a game can be given by the maxmin of the payoff polynomial over all possible values satisfying the strategy constraints.




**(a)** Max **without ALR but has S-ALR** **(b)** Max **with ALR**

**Figure 3: Equivalent ALR game using S-ALR for game without ALR**

*Overview of our work.* In this work, our mantra for simplifying games is to find simpler games with same payoff polynomials (upto renaming of variables). Leaf monomials are the building blocks of payoff polynomials. We give methods to generate from a given game-structure $\mathcal{T}$, a transformed game-structure $\mathcal{T}'$ with A-loss recall such that: either $\mathcal{T}'$ has the same set of leaf monomials (Section 4), or each leaf monomial of $\mathcal{T}$ is a linear combination of the leaf monomials of $\mathcal{T}'$ (Section 5).

## 4 SHUFFLED A-LOSS RECALL

We start with an example. The game-structure in Fig. 3a is an equivalent game of version **II** of the matching-unmatching game (Fig. 1b) obtained by merging die outcome 0 and 1 (and renaming $H, T$), with $p_1 = \frac{2}{3}, p_2 = \frac{1}{3}$. The game does not have ALR: we have $\mathcal{H}_{\text{Max}}(I_3) = \{b, \bar{b}, c, \bar{c}\}$. Since $\{b, \bar{b}\}$ and $\{c, \bar{c}\}$ are from different information sets, the pair of histories $b$ and $c$, for instance, is a witness for no ALR. The player forgets what she knew about Chance actions. Now, consider the game-structure in Fig. 3b, obtained by *shuffling* the actions ($a$ goes above $b$ and $c$). This game-structure has ALR. The crucial observation is that both the game-structures, Fig. 3a and Fig. 3b, lead to the same leaf monomials[3]: $\{x_a x_b,\ x_a x_{\bar{b}},\ x_{\bar{a}} x_b,\ x_{\bar{a}} x_{\bar{b}},\ x_a x_c,\ x_a x_{\bar{c}},\ x_{\bar{a}} x_c,\ x_{\bar{a}} x_{\bar{c}}\}$. Similarly, in Fig. 1b, by shuffling the turns of Alice and Bob, we get an ALR recall game that induces the same leaf monomials.

We say that the game-structure of Fig. 3a has *shuffled A-loss recall.* Even though the game-structure originally does not have ALR, it can be shuffled in some way to get an ALR structure. Not every game-structure has shuffled A-loss recall. In this section, we

[2] Square-Root-Sum is the decision problem of checking if the sum of the square roots of $k$ positive integers is less than another positive number

[3] Two monomials are same if their sets of variables are same

provide a PTIMEalgorithm to identify whether a game-structure has shuffled A-loss recall. If the answer is yes, the algorithm also computes the shuffled game-structure. As a result, we are able to show that one-player shuffled A-loss recall games can be solved in PTIME. We will keep our discussion to one-player games played by Max, and later in Section 6 discuss extensions to two-player games. We will require few notions and notations, which we introduce gradually as we need them.

For a game structure $\mathcal{T}$, we write $L_{\mathcal{T}}$ for the set of its leaves. Define $|\mathcal{T}|$, the *size* of a game structure $\mathcal{T}$, to be $|L_{\mathcal{T}}|$, the total number of its leaves. We work with history sequences originating from game structures. Fix a finite set of information sets $\mathcal{I}$ and a set of actions $\mathrm{Act}(I)$ for each $I \in \mathcal{I}$. Recall that the action sets of distinct information sets are disjoint. Let $A = \biguplus_{I\in\mathcal{I}} \mathrm{Act}(I)$. A *sequence* is a finite word over $A^*$ that contains at most one letter from each $\mathrm{Act}(I)$ [4]. Let $s[i]$ denote the $i$th action in $s$. For sequences $s_1$ and $s_2$ of length $k$, we say $s_2$ is a permutation of $s_1$ if $\exists$ a bijective function $\rho : \{1,\ldots,k\} \mapsto \{1,\ldots,k\}$ such that $\forall i$, $s_1[i] = s_2[\rho(i)]$.

**Definition 2** (Shuffled A-loss recall)**.** *A game structure $\mathcal{T}$ is said to have shuffled A-loss recall (s-alr) if $\exists$ a game structure $\mathcal{T}'$ with $|\mathcal{T}| = |\mathcal{T}'|$ such that*

- *$\mathcal{T}'$ has A-loss recall*
- *There is a bijection $f : L_{\mathcal{T}} \mapsto L_{\mathcal{T}'}$ such that $\forall t \in L_{\mathcal{T}}$, $hist(f(t))$ is a permutation of $hist(t)$.*

When $\mathcal{T}$ has s-alr we call the structure $\mathcal{T}'$ an s-alr witness of $\mathcal{T}$. The following lemma is a consequence on leaf monomials.

**Lemma 1.** *Suppose $\mathcal{T}$ has s-alr with s-alr witness $\mathcal{T}'$, then $\mathcal{T}$ and $\mathcal{T}'$ have the same set of leaf monomials.*

*Detecting s-alr.* Checking whether a structure $\mathcal{T}$ has pfr or alr can be done in polynomial-time, simply by checking histories at every information set. On the other hand, from the definition of s-alr, it is not immediate if one could test it efficiently. One approach could be finding good permutations for each leaf history in order to get the s-alr witness but this could potentially lead to exponentially many checks. In the following discussion we will provide a polynomial-time algorithm to test s-alr in a structure.

**Theorem 1.** *Given a game structure $\mathcal{T}$, there is an algorithm that checks if $\mathcal{T}$ has s-alr in time $O(|\mathcal{T}|)$. Moreover if $\mathcal{T}$ does have s-alr, this algorithm also outputs an s-alr witness structure $\mathcal{T}'$.*

To prove Theorem 1, we will work with the history sequences. For a game structure $\mathcal{T}$, recall that $\mathcal{H}(L_{\mathcal{T}})$ is the set of all leaf histories in $\mathcal{T}$. We will construct a set of leaf histories $H'$ such that $H' = \mathcal{H}(L_{\mathcal{T}'})$ for some alr structure $\mathcal{T}'$. Since we work with sequences, and not game-structures themselves, we will need a notion of alr for sets of words. When we are given a game, alr can be detected by looking at the histories. When we are given a set of histories, it is not as direct. We will need to determine some structure inside the sequences.

*alr on sequence sets.* Two sequences $s_1, s_2$ over $A$ are said to be *connected* if there is some information set $I$ such that both $s_1$ and $s_2$ contain an action from $\mathrm{Act}(I)$. E.g., suppose $\mathrm{Act}(I_1) = \{a, b\}$, $\mathrm{Act}(I_2) = \{c, d\}$, $\mathrm{Act}(I_3) = \{e, f\}$; then $s_1 = ac$ and $s_2 = eb$ are connected since $s_1$ contains $a$, $s_2$ contains $b$, both of which are in $\mathrm{Act}(I_1)$. Sequences $s_1, s_2$ are said to be *disconnected* if they are not connected. For the same alphabet as before, let $s_3 = e$, then $s_1$ and $s_3$ are disconnected.

We say that a set of sequences $S$ is connected if for every disjoint partition of $S$ as $S_1 \uplus S_2$ (where $S_1, S_2$ are non-empty), there exist $s_1 \in S_1$ and $s_2 \in S_2$ such that $s_1$ and $s_2$ are connected. E.g., consider $S = \{ac, eb, e\}$ with information sets as above. This set $S$ is connected, even though $ac$ and $e$ are not connected. For a set $S$, we can construct an undirected graph as follows: each $s \in S$ is a vertex, and there is an edge between $s_1, s_2 \in S$ if they are connected. Notice that a set $S$ is connected iff there is a path between any two vertices in this graph. This interpretation allows to decompose $S$ uniquely as $S = \biguplus_i S_i$ where each $S_i$ is a *maximal connected component* in the graph.

We can now give a definition of alr on sequences. This is defined inductively as follows. The set $\{\epsilon\}$ has alr. A disconnected set $S$ with decomposition $\biguplus_i S_i$ has alr if each of its connected components $S_i$ has alr. A connected set $S$ has alr if there exists an $I \in \mathcal{I}$ s.t.:

(1) every sequence in $S$ starts with $\mathrm{Act}(I)$: i.e. each $w \in S$ is of the form $au$ for some $a \in \mathrm{Act}(I)$, and
(2) the set of continuations of each $a \in \mathrm{Act}(I)$ has alr: for each $a \in \mathrm{Act}(I)$, the set $\{u \mid au \in S\}$ has alr.

Let us illustrate this definition on examples from Fig. 2. In 2a, we have the leaf histories $H_1 = \{ac, ad, be, bf\}$. Notice that $H_1$ is a connected set. There is an information set $I_1$ with $\mathrm{Act}(I_1) = \{a, b\}$ such that the first condition above is true. For the second condition, let us look at the continuations: $H_1^a = \{c, d\}$ and $H_1^b = \{e, f\}$. Both $H_1^a$ and $H_1^b$ are connected and satisfy the first condition. The second condition is vacuously true for $H_1^a$ and $H_1^b$. This shows $H_1$ has alr (as expected, since on game-structures, pfr is a subclass of alr). Now, let us look at Fig. 3a. The leaf histories are given by $H_2 = \{ba, b\bar{a}, \bar{b}a, \bar{b}\bar{a}, ca, c\bar{a}, \bar{c}a, \bar{c}\bar{a}\}$. Observe that $H_2$ is connected. However, the first condition in the alr definition does not hold. So $H_2$ is not alr. We can show that the recursive definition of alr-sets and that of alr game structures are equivalent.

**Proposition 1.** *A game structure $\mathcal{T}$ has alr iff $\mathcal{H}(L_{\mathcal{T}})$ has alr.*

*s-alr on sequence sets.* We can also extend the definition of s-alr to sequence sets. $S$ has s-alr if $\exists$ another set $S'$ such that (i) $S'$ has alr and (ii) there is a bijection $f : S \mapsto S'$ where $\forall s \in S$, $f(s)$ is a permutation of $s$. We call the set $S'$ an s-alr witness of $S$.

Exploiting the recursive definition of alr sets we will provide recursive necessary and sufficient conditions for sequence sets to have s-alr. Firstly, one can check for s-alr for a set $S$ by checking s-alr for each individual maximal connected components.

**Proposition 2.** *Let $S$ be a disconnected set and $S = \biguplus_i S_i$ be the decomposition of $S$ into maximal connected components. Then $S$ has s-alr iff $\forall i, S_i$ has s-alr .*

**Corollary 1.** *Let $S = \uplus_i S_i$ be the decomposition of $S$ where each $S_i$ has s-alr witnessed by sets $S_i'$. Then $S$ has s-alr and $S' = \bigcup_i S_i'$ is an s-alr witness of $S$.*

[4] We restrict to such words in $A^*$ since histories in an nam game structure have this property

Secondly, for connected sets we will provide another recursive condition to have s-ALR. We use another notation. For an action $a \in A$, we write $S_a := \{u_1 u_2 \mid u_1 a u_2 \in S\}$, for the set of sequences obtained by picking a sequence in $S$ that contains $a$, and dropping the $a$ from it.

**Proposition 3.** *Let $S$ be a connected set. $S$ has s-ALR iff there exists an information set $I \in \mathcal{I}$ such that*

(1) *every sequence $s \in S$ contains an action from* $\mathrm{Act}(I)$,
(2) *and for all $a \in \mathrm{Act}(I)$, the set $S_a$ has s-ALR.*

The above proposition can be naturally translated into an algorithm. However, this does not yet ensure a polynomial-time complexity. If there are two information sets $I_1$ and $I_2$ that satisfy Condition 1 of Proposition 3, the order in which we pick them might, in principle, create a difference and hence one has to guess the right order. The next lemma says this does not happen.

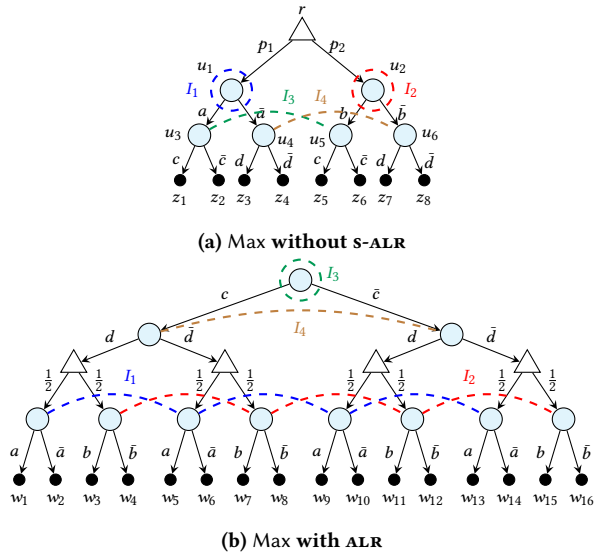
**Lemma 2.** *Let $S$ be a connected set and let $I$ be an arbitrary information set such that every sequence $s \in S$ contains an action from* $\mathrm{Act}(I)$. *Then: $S$ has s-ALR iff for all $a \in \mathrm{Act}(I)$, the set $S_a$ has s-ALR. Moreover, if for each $a$, $S'_a$ is s-ALR witness of $S_a$, then $\bigcup_a aS'_a$ is an s-ALR witness of $S$.*

**Algorithm 1** Compute shuffled A-loss recall

**Input** : $S$
**Output**: s-ALR witness $S'$ of $S$ if it exists
**if** $S$ is connected **then**
  **if** $\exists I$ such that every $s \in S$ contains an action from $\mathrm{Act}(I)$ **then**
    **for** $a \in \mathrm{Act}(I)$ **do**
      $S'_a \leftarrow$ s-ALR witness of $S_a$
    **end for**
    **return** $\bigcup_a aS'_a$
  **else**
    **EXIT** and report $S$ does not have s-ALR
  **end if**
**else**
  $S = \uplus S_i$ where each $S_i$ is connected
  $S'_i \leftarrow$ s-ALR witness of $S_i$
  **return** $\cup_i S'_i$
**end if**

Proposition 3 and Lemma 2 lead to Algorithm 1.This algorithm runs in time $O(|S|)$, proving Theorem 1.

Here is an example run of the algorithm on Fig. 3a. We have $S = \{ba, b\bar{a}, \bar{b}a, \bar{b}\bar{a}, ca, c\bar{a}, \bar{c}a, \bar{c}\bar{a}\}$. All the sequences contain an action from $\mathrm{Act}(I_1) = \{a, \bar{a}\}$. For the recursive call, we the set $H = \{b, \bar{b}, c, \bar{c}\}$. This set is disconnected, with components $H_1 = \{b, \bar{b}\}$ and $H_2 = \{c, \bar{c}\}$. For $H_1, H_2$ the same sets are witness for s-ALR. For $H$, the witness is the same set again. For $S$, the witness is $aH \uplus \bar{a}H = \{ab, a\bar{b}, \ldots, \bar{a}c, \bar{a}\bar{c}\}$. This set can be translated to the game Fig. 3b, with ALR. Now, back to our running example Fig. 1b. Let $H_{01}, T_{01}$ and $H_2, T_2$ be Alice's actions out of the information sets obtained after die roll 0 or 1, and 2 respectively. The induced sequence set is $\{H_{01}H, H_{01}T, T_{01}H, T_{01}T, H_2H, T_2T\}$, which can be seen to have s-ALR. In fact, for all values for $n$, the game **II** has s-ALR.




**(a)** Max **without s-ALR**

**(b)** Max **with ALR**

**Figure 4: Equivalent ALR game using ALR-span for game without s-ALR**

Theorem 1 and the fact that ALR games can be solved in polynomial-time give us the following theorem, and hence a new polynomial-time solvable class with imperfect recall.

**Theorem 2.** *The maxmin value in one-player games with s-ALR can be computed in polynomial time.*

## 5 SPAN

We move on to another way of simplifying game-structures by generalizing s-ALR. The game-structure in Fig. 4a (call it $\mathcal{T}_1$) is an equivalent version of game **III** (Fig. 1c). It neither has PFR, nor ALR. Using Algorithm 1, we can show that it does not have s-ALR either. Now, consider the game-structure in Fig. 4b (call it $\mathcal{T}_1'$). It has ALR. Each leaf monomial of $\mathcal{T}_1$ can be written as a linear combination of the leaf monomials of $\mathcal{T}_1'$: e.g., the leaf monomial $x_a x_{\bar{c}}$ of $\mathcal{T}_1$ is equal to $x_{\bar{c}} x_d x_a + x_{\bar{c}} x_{\bar{d}} x_a$, a combination of leaf monomials of $\mathcal{T}_1'$. The game-structure $\mathcal{T}_1$ is said to be *spanned by* $\mathcal{T}_1'$. This property allows to solve games derived from the structure $\mathcal{T}_1$ by converting them into a game on $\mathcal{T}_1'$ with a suitably designed utility function so that both games induce the same payoff polynomial, and then solving the resulting A-loss recall game. This is illustrated in Fig. 4a and Fig. 4b. Results in this section:

- We show that every NAM imperfect recall game structure is spanned by an ALR structure (Theorem 3). The caveat is that the smallest ALR span may be of exponential size: we exhibit a family of game structures where this happens (Theorem 4).
- We provide an algorithm to compute an A-loss recall span of smallest size. We show that the associated decision problem is in NP (Theorem 5). We also identify classes of games with 'small' ALR-span using a new parameter (Corollary 2).

We will now formally present ALR-span. Similar to last section, we will keep our discussion to one-player games and later discuss extensions to two players in Section 6.

Since we will deal with polynomials formed using leaf monomials, we need the notion of reducing one polynomial to another. Recall that since these variables are denoting behavioral strategies, every valuation to the variables satisfies the strategy constraints. We say polynomial $f_1$ reduces to $f_2$ under strategy constraints if we can get $f_2$ by applying finitely many substitutions in $f_1$ of the form $\sum_{a \in Act(I)} x_a = 1$. E.g., $x_a x_d x_c + x_a x_d x_{\bar{c}} + x_{\bar{a}} x_d x_c + x_{\bar{a}} x_d x_{\bar{c}}$ reduces to $x_d$ by applying the substitutions $x_a + x_{\bar{a}} = 1$ and $x_c + x_{\bar{c}} = 1$. Observe that, when $f_1$ reduces $f_2$, they are essentially the same polynomials over the space with strategy constraints, i.e. they evaluate to the same value under every assignment of values satisfying the strategy constraints.

**Definition 3** (ALR-span). *Let $\mathcal{T}$ be a game structure with set of leaf monomials $X(\mathcal{T})$. We call a structure $\mathcal{T}'$ an ALR-span of $\mathcal{T}$ if*

- *$\mathcal{T}'$ has ALR.*
- *each monomial in $X(\mathcal{T})$ can be generated by monomials in $X(\mathcal{T}')$ by linear combinations i.e. $\forall \mu \in X(\mathcal{T})$ there exist coefficients $\{c^{\mu}_{\mu'}\}_{\mu' \in X(\mathcal{T}')} \in \mathbb{R}^{|X(\mathcal{T}')|}$ such that the polynomial $\sum_{\mu' \in X(\mathcal{T}')} c^{\mu}_{\mu'} \mu'$ reduces to $\mu$ under strategy constraints.*

The game structure $\mathcal{T}_1'$ in Fig. 4b is an ALR-span of $\mathcal{T}_1$ in Fig. 4a. E.g., for the monomial $x_{\bar{b}} x_{\bar{d}}$ in $\mathcal{T}_1$, the linear combination $x_c x_{\bar{d}} x_{\bar{b}} + x_{\bar{c}} x_{\bar{d}} x_{\bar{b}}$ reduces to it by substituting $x_d + x_{\bar{d}} = 1$. In fact, observe that for any monomial $\mu$ in $X(\mathcal{T}_1)$, the sum of the two monomials in $X(\mathcal{T}_1')$ that contain the actions in $\mu$, reduces to $\mu$.

The next proposition states that we can use ALR-span $\mathcal{T}_1'$ to solve games on structure $\mathcal{T}_1$ (see Fig. 4a and Fig. 4b). We can assign suitable payoffs $w_i$s in terms of $z_i$s and $p_i$s to get an equivalent game. Let $t_i$ be the leaf that has payoff $w_i$.The payoff $w_i$ is the sum of the quantities $\frac{\text{Prob}_{\text{Chance}}(t) c^{\mu(t)}_{\mu(t_i)} \mathcal{U}(t)}{\text{Prob}_{\text{Chance}}(t_i)}$ for all $t$ in $\mathcal{T}_1$, in which $\mu(t_i)$ contributed to generate $\mu(t)$. E.g., $t_1$, with polynomial $x_c x_d x_a$ contributes in generating only $x_a x_c$ (for the leaf corresponding to $z_1$). Hence $w_1 = 2p_1 z_1$. Similarly, $w_5 = 2p_1 z_1$. In the payoff polynomial of the second game, since $x_c x_d x_a + x_c x_{\bar{d}} x_a = x_a x_c$, we will have the term $p_1 z_1 x_a x_c$ in the payoff polynomial. This way, we will get the same payoff polynomial as original game, thus reducing the original game to a game on $\mathcal{T}_1'$.

**Proposition 4.** *Let $\mathcal{T}$ be a game structure and $\mathcal{T}'$ an ALR-span of $\mathcal{T}$. Then for every game $G = (\mathcal{T}, \delta, \mathcal{U})$ on $\mathcal{T}$, there exists a game $G' = (\mathcal{T}', \delta', \mathcal{U}')$ on $\mathcal{T}'$ such that solving $G$ can be reduced to solving $G'$.*

*Existence of ALR-spans.* From the definition, one can see that when $\mathcal{T}$ has S-ALR, an S-ALR witness is also an ALR-span of $\mathcal{T}$. Surprisingly, we can show that every NAM structure has ALR-span.

**Theorem 3.** *Every NAM-game structure $\mathcal{T}$ has an ALR-span.*

Proof Sketch. We can explicitly provide an ALR-span $\mathcal{T}'$ of $\mathcal{T}$. Let $\mathcal{I}$ be the set of information sets in $\mathcal{T}$. The structure $\mathcal{T}'$ has $|\mathcal{I}|$ levels of player nodes corresponding to each $I \in \mathcal{I}$. All nodes in a level are placed in one information set. Therefore the leaves of $\mathcal{T}'$ are all possible monomials using the information sets $\mathcal{I}$. Moreover, $\mathcal{T}'$ has ALR. To generate a monomial $\mu \in X(\mathcal{T})$ one can combine the monomials of all paths containing actions from $\mu$ in $X(\mathcal{T}')$. □

*Minimal ALR-span.* In order to take advantage of Proposition 4, one would need to find small ALR-spans. We observe that the ALR-span obtained in the proof of Theorem 3 has exponential size. We will now delve into finding ALR-spans of smallest size : a *minimal ALR-span*. First we will list some key observations concerning minimal ALR-spans which will lead to an algorithm for computing one. Then we will show that the exponential blowup in size of ALR-spans is unavoidable in general by exhibiting a class of games with minimal ALR-span of exponential size. Since ALR-games are solvable in polynomial time, this aligns with the fact that the maxmin problem for general one-player NAM-games is NP-hard.

Similar to Section 4, we will work directly with sequence sets. The notions related to span are extended to sequence sets in a natural manner, i.e. when $\mathcal{T}'$ is an ALR-span of $\mathcal{T}$, $\mathcal{H}(L_{\mathcal{T}'})$ is an ALR-span of $\mathcal{H}(L_{\mathcal{T}})$

**Proposition 5.** *For a sequence set $S$, let $S = \biguplus_i S_i$ be the decomposition of $S$ into maximal connected components. Let $S_i'$ be a minimal ALR-span of $S_i$. Then $S' = \biguplus_i S_i'$ is a minimal ALR-span of $S$*

We can show that a minimal ALR-span of a connected set is also connected. By definition, the ALR-span is a set of sequences that has ALR, and by the first point in the definition of ALR on sequence sets, we deduce that there is an $I$ such that all sequences in the minimal span start with $\text{Act}(I)$.

**Lemma 3.** *Let $S$ be a connected set and $S'$ be a minimal ALR-span of $S$. Then $\exists I \in \mathcal{I}$, such that all sequences in $S'$ start with $\text{Act}(I)$.*

The next observation says how to find this $I$.

**Lemma 4.** *Let $S$ be a connected set. If there is an $I$ such that every sequence of $S$ has an action in $\text{Act}(I)$, then there is a minimal ALR-span $S'$ of $S$ such that all sequences in $S'$ start with $\text{Act}(I)$.*

If there is no such $I$, then we need to enumerate over all information sets to find the smallest. Once we fix an $I$, the next lemma says how to find a minimal ALR-span which starts with $\text{Act}(I)$.

**Lemma 5.** *Let $S$ be a connected set, and let $I \in \mathcal{I}$. An ALR-span of smallest size among all ALR-spans starting with $\text{Act}(I)$ is the following: $S' = \bigcup_{a \in \text{Act}(I)} aH_a'$ where $H_a'$ is a minimal ALR-span of $S_a \cup S_{\bar{I}}$ where $S_{\bar{I}} := \{s \in S \mid s$ contains no action from $\text{Act}(I)\}$.*

*Algorithm for computing minimal ALR-span.* Based on the lemmas above, we can design a recursive algorithm to compute a minimal ALR-span for an input sequence set $S$. Firstly, if $S$ is disconnected, based on Proposition 5, $S$ is decomposed into maximal connected components and a minimal ALR-span is computed for each component. When $S$ is connected: (1) for each $I \in \mathcal{I}$, compute $H_I = \{s \mid s$ contains no action from $\text{Act}(I)\}$; (2) if there is some $I$ such that $H_I = \emptyset$ (Lemma 4), find the smallest ALR-span starting with $\text{Act}(I)$ using Lemma 5; (3) else, for each $I$, compute the smallest ALR-span starting from $\text{Act}(I)$ using Lemma 5 and return the smallest. We remark that in the algorithm to find the minimal ALR-span, $\epsilon$ might appear in intermediate sequence sets. In that case, we just remove $\epsilon$ from the set and continue the algorithm.

Next, we show that there are game-structures whose minimal ALR-spans are exponentially large.

**Theorem 4.** *For every $n > 0$, there exists a game structure $\mathcal{T}_n$ of size $O(n^2)$ such that the size of a minimal ALR-span of $\mathcal{T}_n$ is $\Omega(2^n)$.*

*Complexity.* In order to investigate the complexity of computing a minimal ALR-span of a game structure, we consider the following decision problem and show it to be in NP. We leave open the question of whether it is NP-hard.

MIN-ALR-SPAN: Given a game structure $\mathcal{T}$ and an integer $k > 0$, is there a game structure $\mathcal{T}'$ such that $|\mathcal{T}'| \leq k$ and $\mathcal{T}'$ is an ALR-span of $\mathcal{T}$?

**Theorem 5.** *The decision problem MIN-ALR-SPAN is in* NP.

Proof Sketch. One can guess a $\mathcal{T}'$ of size at most $k$, and also the linear combinations required for each $\mu \in X(\mathcal{T})$. This is polynomial in size. For each monomial $\mu \in X(\mathcal{T})$ we verify if the monomials in $X(\mathcal{T}')$ containing all variables from $\mu$ can generate $\mu$. This can be done efficiently in polynomial-time. □

*Efficiently solvable classes.* We get back to finding efficiently solvable classes of imperfect recall games, this time using a parameter called *shuffle-depth SD*, that is naturally derived from our algorithm for computing a minimal ALR-span.

When $S$ is disconnected with $S = \uplus S_i$ and each $S_i$ connected : define SD(S) = $\max_i SD(S_i)$. When $S$ is connected and $S$ has S-ALR, define SD(S) = 0. Otherwise define

$$SD(S) = 1 + \min_{I} \max_{a \in I} \mathrm{SD}(S_a \cup S_{\bar{I}})$$

For a $\mathcal{T}$ with SD $k$, our minimal ALR-span algorithm would encounter sets with S-ALR after recursively running upto depth $k$. This takes $O(|\mathcal{T}|^{k+1})$ time.

**Proposition 6.** *The minimal ALR-span of game structure $\mathcal{T}$ with SD = $k$ can be computed in time $O(|\mathcal{T}|^{k+1})$.*

**Corollary 2.** *The maxmin value in a one-player game can be computed in* PTIME *for games having structures with constant SD.*

The game **III** for any $n$ has SD 2, and using our algorithm one can find equivalent ALR-game with just a quadratic blowup in size.

## 6 TWO-PLAYER GAMES

For a two-player game structure $\mathcal{T}$ and the corresponding sequence set $S$, we can look at the projection of sequences on individual player actions, $S_{\mathsf{Max}}$ and $S_{\mathsf{Min}}$, consider their ALR-spans and knit them together. Consider the 2-player game in Fig. 5. Projections of the sequence set in this game w.r.t individual players would correspond to game structures in Fig. 5b (for Max) and Fig. 3a (for Min). The structure Fig. 5b already has PFR. As seen before, Fig. 3b is an S-ALR witness for Fig. 3a. We can plug in this S-ALR witness to all leaf nodes of Fig. 5b and get the structure Fig. 5c. Information sets are maintained across all copies of Fig. 5c as shown in the illustration. More generally, if $\mathcal{T}'_{\mathsf{Max}}$ is an ALR-span of $\mathcal{T}_{\mathsf{Max}}$, and $\mathcal{T}'_{\mathsf{Min}}$ is an ALR-span of $\mathcal{T}_{\mathsf{Min}}$, we can obtain an (ALR, ALR) structure where all nodes of Max precede all nodes of Min, and a copy of $\mathcal{T}'_{\mathsf{Min}}$ is attached to each leaf node of Max, and information sets of Min are maintained across all copies of $\mathcal{T}'_{\mathsf{Min}}$, as shown in Fig. 5.

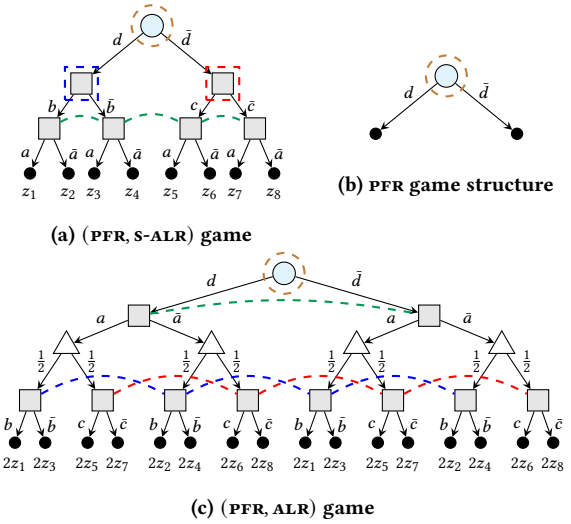



**Figure 5: Equivalent two player game obtained by composing ALR-spans of respective** Max, Min **structures**

We can assign suitable payoffs in a similar manner to Proposition 4 to get an equivalent game on this new game structure.

**Theorem 6.** *Solving an (NAM, NAM) game on structure $\mathcal{T}$ can be reduced to solving an (ALR, ALR) game on a structure of size $|\mathcal{T}'_{\mathsf{Max}}||\mathcal{T}'_{\mathsf{Min}}|$ where $\mathcal{T}'_{\mathsf{Max}}$ and $\mathcal{T}'_{\mathsf{Min}}$ are minimal ALR-spans of $\mathcal{T}_{\mathsf{Min}}$.*

Since (PFR, ALR)-games can be solved in polynomial-time, Theorem 6 and Corollary 2 lead to new polynomial-time solvable classes.

**Corollary 3.** *The maxmin value in a (PFR, NAM) game where SD of $\mathcal{T}_{\mathsf{Min}}$ is constant can be computed in polynomial time. As a conseqeunce, (PFR, S-ALR) games can be solved in polynomial time.*

## 7 CONCLUSION

We have presented a study of imperfect recall without absentmindedness, through the lens of A-loss recall. Specifically, we have given two methods to transform imperfect recall games to A-loss recall games. Behavioral strategies for the original game can be obtained by analyzing the transformed game. This investigation has resulted in new PTIME solvable classes of one-player and two-player games. We have also shown how to find a transformation of minimal size. It would be interesting to see the influence of these notions of S-ALR and ALR-span, and the idea of using sequence sets instead of games, in algorithms that use imperfect recall abstractions.

In summary, in this work, we have laid the foundations to simplify imperfect recall in terms of ALR. We do hope that this perspective leads to further theoretical and experimental investigations.

## ACKNOWLEDGMENTS

The second author is supported by the EPSRC through project EP/X03688X/1. A considerable part of this work was done when the second author was at LaBRI, Université de Bordeaux and then at IRIF, Université Paris Cité.

## REFERENCES


[1] Noam Brown, Sam Ganzfried, and Tuomas Sandholm. 2015. Hierarchical Abstraction, Distributed Equilibrium Computation, and Post-Processing, with Application to a Champion No-Limit Texas Hold'em Agent. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind (Eds.). ACM, 7–15. http://dl.acm.org/citation.cfm?id=2772885

[2] Noam Brown and Tuomas Sandholm. 2017. Libratus: The Superhuman AI for No-Limit Poker. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 5226–5228. https://doi.org/10.24963/ijcai.2017/772

[3] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890. https://doi.org/10.1126/science.aay2400

[4] Andrea Celli and Nicola Gatti. 2018. Computational Results for Extensive-Form Adversarial Team Games. In *Proceedings of the Thirty-Second Conference on Artificial Intelligence (AAAI'18)*. AAAI Press, New Orleans, Louisiana, USA, 965–972.

[5] Jirí Cermák, Branislav Bosanský, Karel Horák, Viliam Lisý, and Michal Pechoucek. 2018. Approximating maxmin strategies in imperfect recall games using A-loss recall property. *Int. J. Approx. Reasoning* 93 (2018), 290–326.

[6] Jiri Cermak, Branislav Bosanský, and Viliam Lisý. 2017. An Algorithm for Constructing and Solving Imperfect Recall Abstractions of Large Extensive-Form Games. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 936–942. https://doi.org/10.24963/IJCAI.2017/130

[7] Vincent Conitzer. 2019. Designing Preferences, Beliefs, and Identities for Artificial Intelligence. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 9755–9759. https://doi.org/10.1609/AAAI.V33I01.33019755

[8] Sam Ganzfried and Tuomas Sandholm. 2014. Potential-Aware Imperfect-Recall Abstraction with Earth Mover's Distance in Imperfect-Information Games. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, Carla E. Brodley and Peter Stone (Eds.). AAAI Press, 682–690. https://doi.org/10.1609/AAAI.V28I1.8816

[9] Hugo Gimbert, Soumyajit Paul, and B. Srivathsan. 2020. A Bridge between Polynomial Optimization and Games with Imperfect Recall. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20*. International Foundation for Autonomous Agents and Multiagent Systems.

[10] Hugo Gimbert, Soumyajit Paul, and B. Srivathsan. 2025. Simplifying imperfect recall games. arXiv:2502.13933 [cs.GT] https://arxiv.org/abs/2502.13933

[11] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. 2013. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. 271–278.

[12] Mamoru Kaneko and J Jude Kline. 1995. Behavior strategies, mixed strategies and perfect recall. *International Journal of Game Theory* 24 (1995), 127–145.

[13] J Jude Kline. 2002. Minimum memory for equivalence between ex ante optimality and time-consistency. *Games and Economic Behavior* 38, 2 (2002), 278–305.

[14] Daphne Koller and Nimrod Megiddo. 1992. The complexity of two-person zero-sum games in extensive-form. *Games and Economic Behavior* 4, 4 (1992), 528–552.

[15] Christian Kroer and Tuomas Sandholm. 2016. Imperfect-Recall Abstractions with Bounds in Games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, Vincent Conitzer, Dirk Bergemann, and Yiling Chen (Eds.). ACM, 459–476. https://doi.org/10.1145/2940716.2940736

[16] Nicolas S. Lambert, Adrian Marple, and Yoav Shoham. 2019. On equilibria in games with imperfect recall. *Games and Economic Behavior* 113 (2019), 164–185. https://doi.org/10.1016/j.geb.2018.09.007

[17] Marc Lanctot, Richard G. Gibson, Neil Burch, and Michael Bowling. 2012. No-Regret Learning in Extensive-Form Games with Imperfect Recall. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress. http://icml.cc/2012/papers/58.pdf

[18] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. 2017. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356, 6337 (2017), 508–513. https://doi.org/10.1126/science.aam6960

[19] Bernhard Von Stengel. 1996. Efficient Computation of Behavior Strategies. *Games and Economic Behavior* 14, 2 (1996), 220–246.

[20] Emanuel Tewolde, Caspar Oesterheld, Vincent Conitzer, and Paul W. Goldberg. 2023. The computational complexity of single-player imperfect-recall games *(IJCAI '23)*. Article 321, 10 pages. https://doi.org/10.24963/ijcai.2023/321

[21] Emanuel Tewolde, Brian Hu Zhang, Caspar Oesterheld, Manolis Zampetakis, Tuomas Sandholm, Paul Goldberg, and Vincent Conitzer. 2024. Imperfect-Recall Games: Equilibrium Concepts and Their Complexity. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, Kate Larson (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2994–3004. https://doi.org/10.24963/ijcai.2024/332 Main Track.

[22] Bernhard von Stengel and Daphne Koller. 1997. Team-Maxmin Equilibria. *Games and Economic Behavior* 21, 1 (1997), 309–321. https://doi.org/10.1006/game.1997.0527

[23] Kevin Waugh, Martin Zinkevich, Michael Johanson, Morgan Kan, David Schnizlein, and Michael H. Bowling. 2009. A Practical Use of Imperfect Recall. In *Eighth Symposium on Abstraction, Reformulation, and Approximation (SARA'09)*. AAAI, California, USA, 175 – 182.

[24] Kevin Waugh, Martin Zinkevich, Michael Johanson, Morgan Kan, David Schnizlein, and Michael H. Bowling. 2009. A Practical Use of Imperfect Recall. In *Eighth Symposium on Abstraction, Reformulation, and Approximation, SARA 2009, Lake Arrowhead, California, USA, 8-10 August 2009*, Vadim Bulitko and J. Christopher Beck (Eds.). AAAI. http://www.aaai.org/ocs/index.php/SARA/SARA09/paper/view/839

[25] Ernst Zermelo. 1913. Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. InProceedings of the Fifth International Congress of Mathematicians II.

[26] Jiří Čermák, Viliam Lisý, and Branislav Bošanský. 2020. Automated construction of bounded-loss imperfect-recall abstractions in extensive-form games. *Artificial Intelligence* 282 (2020), 103248. https://doi.org/10.1016/j.artint.2020.103248