Bounded-Memory Strategies in Partial-Information Games

Sougata Bose
sougata.bose@liverpool.ac.uk
University of Liverpool
Liverpool, UK

Rasmus Ibsen-Jensen
r.ibsen-jensen@liverpool.ac.uk
University of Liverpool
Liverpool, UK

Patrick Totzke
totzke@liverpool.ac.uk
University of Liverpool
Liverpool, UK

ABSTRACT

We study the computational complexity of solving stochastic games with mean-payoff objectives. Instead of identifying special classes in which simple strategies are sufficient to play ε-optimally, or form ε-Nash equilibria, we consider general partial-information multiplayer games and ask what can be achieved with (and against) finite-memory strategies up to a given bound on the memory.

We show NP-hardness for approximating zero-sum values, already with respect to memoryless strategies and for 1-player reachability games. On the other hand, we provide upper bounds for solving games of any fixed number of players k. We show that one can decide in polynomial space if, for a given k-player game, $\varepsilon \geq 0$ and bound b, there exists an ε-Nash equilibrium in which all strategies use at most b memory modes.

For given $\varepsilon > 0$, finding an ε-Nash equilibrium with respect to b-bounded strategies can be done in $\mathsf{FNP}^{\mathsf{NP}}$. Similarly for 2-player zero-sum games, finding a b-bounded strategy that, against all b-bounded opponent strategies, guarantees an outcome within ε of a given value, can be done in $\mathsf{FNP}^{\mathsf{NP}}$. Our constructions apply to parity objectives with minimal simplifications.

Our results improve the status quo in several well-known special cases of games. In particular, for 2-player zero-sum concurrent mean-payoff games, one can approximate ordinary zero-sum values (without restricting admissible strategies) in $\mathsf{FNP}^{\mathsf{NP}}$.

CCS CONCEPTS

• **Theory of computation → Algorithmic game theory**; **Exact and approximate computation of equilibria**.

KEYWORDS

Finite-memory strategies, Equilibria, Approximation, Imperfect information games

1 INTRODUCTION

We study stochastic games of infinite duration, where k players jointly move a pebble through a finite directed graph. In every round, all players independently select, possibly at random, one out of finitely many actions. The pebble is moved according to a fixed distribution for the current state and chosen action vector, the players receive an associated reward and the next round begins. In general partial-information games, players do not observe the choices or rewards of other players nor the current state of the pebble. Instead, at the beginning of each round, they receive a signal that may or may not include that information. This subsumes classical settings such as perfect-information games (signals include the current state, rewards, and actions of others), turn-based games (in every round only one player's action affects the outcome), and non-stochastic games (all distributions are Dirac). We focus on games with mean-payoff (aka limit-average, where players aim to maximise their expected long-term average rewards). We discuss in Section 8.1 how our constructions can be applied to parity games as well with minor simplifying adjustments.

We want to solve such games, i.e., compute (ε)-Nash equilibria, zero-sum values and witnessing strategies. However, partial-information games are very expressive and not guaranteed to have (Nash, or ε-Nash) equilibria or zero-sum values [39]. Moreover, already for rather restricted subclasses that do admit ε-optimal strategies, finding them is undecidable: this is the case for single-player, reachability games with a unique signal, aka partially observable Markov decision processes (POMDPs) [14, 29]. Here, finding the best finite-memory strategy (one that is representable by a finite automaton) without providing a bound on its size up front is complete for the recursive-enumerable problems [14]. For the case of partial-information stochastic parity games, where one player has perfect-information, finding the best finite-memory strategy is EXPTIME-complete [12]. Even for perfect-information games such as the classic "Big Match" [4, 21], finite-memory strategies can be much worse than general ones [25]. For this reason, and motivated by applications, much work has focused on identifying special cases of games in which simple strategies are sufficient to play ε-optimally, or to guarantee the existence of ε-Nash equilibria [9, 10, 15, 19, 21, 22, 26, 28, 30, 31, 33, 34].

In this paper, we study the complexity of solving games *with respect to bounded-memory strategies*, meaning that all players are required to play according to a finite-memory strategy where the number of distinct memory modes is bounded a priori. We ask what outcomes players can still guarantee, and how, if one assumes that only strategies up to the given bound are admissible. This approach lets us study "realistic" scenarios that require bounded representations for strategies. Moreover, for many classes of games, values/equilibria with respect to b-bounded strategies coincide with ordinary zero-sum values/equilibria. For example, concurrent reachability games [20, 24, 26, 27] admit ε-optimal strategies that are memoryless ($b = 1$). Consequently, our upper bounds directly apply to such special cases.

There remain significant challenges for solving games even with known bounds on the number of states and memory modes in admissible strategies: (1) optimal strategies/Nash-equilibria might not exist [17], (2) values/outcomes of equilibria can be irrational [16] and can therefore at best be approximated, and (3) strategies with rational distributions may require double-exponentially small probabilities to be (ε-)optimal [27]. This last point is especially problematic since it means it requires more than polynomial time to guess/write down a good strategy with binary encoded values.

Contributions. We provide new complexity bounds for approximating ε-optimal strategies and ε-equilibria in partial-information

games with bounded strategies. We give an overview here; precise claims are delayed to Section 3 after introducing notations.

First, we give NP and coNP lower bounds for approximating the value achievable by bounded strategies under very weak assumptions. More precisely, we show that it is NP-hard to check if in a given 1-player reachability game the value achieved by memoryless strategies exceeds a given threshold. This holds even if the threshold is unary encoded, and even though the games we construct admit optimal memoryless strategies that do not require randomisation. As a simple corollary, we obtain coNP-hardness for 2-player, zero-sum games. An NP lower bound for checking the existence of a memoryless winning strategy for 2-player, partial-information, non-stochastic games was shown by Amoussou-Guenou et al.[1]. Our result uses stochasticity but works for any bounded-memory strategies. Moreover, the NP-hardness holds already for the 1-player case.

As a second and main contribution, we propose a framework that yields simple approximation algorithms low in the polynomial hierarchy (at level ≤ 2). These ultimately work by guessing polynomial representations of witnessing strategies and then approximating outcomes in the resulting Markov chains. Inspired by [20], we use fixed precision floating-point notations to polynomially represent and manipulate doubly-exponentially small values in probability distributions.

The key new ingredient is an iterative state-space reduction scheme for Markov chains. By way of encoding this in FO($\mathbb{R}$), the first-order theory of the reals[2], we derive (1) a PSPACE algorithm for checking the existence, and computing representations of, ε-equilibria for $\varepsilon \geq 0$; (2) that if suitable equilibria/strategies exist for $\varepsilon > 0$, then also ones that are polynomially representable and verifiable; (3) that the loss of precision incurred by implementing the reduction scheme approximately can be bounded. The latter two points use a fixed-precision floating-point representation allowing us to represent doubly-exponentially small values in strategies using a polynomial number of bits.

Our results have direct consequences for solving several well-known types of games including partial-information parity and mean-payoff games [28], stay-in-a-set games [34], quitting games [37] and concurrent reachability, Büchi and parity games [5–8]. In particular, we show how to approximate values/equilibria for 2-player, zero-sum concurrent mean-payoff games (not discounted and with no assumptions on strategies - in particular, allowing infinite memory ones as is sometimes required) [13, 26, 27, 31, 32]. Our algorithm is in $\mathsf{FNP}^{\mathsf{NP}}$, improving on the state-of-the-art algorithms that use polynomial space based on reductions to the existential theory of the reals [16, 26].

Paper structure. We start by introducing notations in Section 2 before stating our results formally in Section 3. In Section 4 we give NP- and coNP-lower bounds. Section 5 presents a value-preserving state-space reduction procedure for Markov Chains that forms the basis for our upper bounds.

In Section 6, we show how to express games and questions about strategies, in the first-order theory of the reals and derive PSPACE upper bounds for checking the existence of ε-equilibria and good zero-sum strategies. In Section 7 we present our main approximation scheme and the resulting upper bounds. This relies on an encoding into FO($\mathbb{R}$) to derive polynomial bounds on the representations of witnessing strategies and an argument that small perturbations of strategies lead to only small changes in values (Section 7.2). In Section 7.3 we show how to approximate values in the Markov chains resulting from fixing candidate strategy profiles. In Section 7.4 we tie the above together into an algorithm to approximate values or equilibria in $\mathsf{FNP}^{\mathsf{NP}}$. We conclude by discussing applications and new results for related classes of games in Section 8.

2 NOTATIONS

For a set X, we write X^k for its k-fold Cartesian product and $a[i]$ for the ith component of a vector $a \in X^k$. Let X^* and X^ω denote the sets of finite and infinite sequences with elements in X, respectively. We write $\mathbb{D}(X)$ for the set of all probability distributions over X.

Partial-information Games. A k-player game is given by finite sets of states V, actions A, signals S, rewards C, and a transition function $\Delta : V \times A^k \to \mathbb{D}(C^k \times S^k \times V)$ that maps the current state and action vector to a probability distribution over triples describing rewards, next signals, and the next state. The game is played in stages $i \in \{0, 1, 2, \dots\}$, in each of which every player receives a signal, selects an action and receives a reward. At stage i, the game is in a vertex v_i. Each player j is sent the signal $s_i[j]$ and selects an action $a_i[j]$. Then a triple of rewards, signals, and successor state $(c_{i+1}, s_{i+1}, v_{i+1})$ is drawn randomly according to $\Delta(v_i, a_i)$ and the game moves to the next stage in vertex v_{i+1}. This random process goes on forever and describes an infinite sequence $\rho = (v_0, s_0, a_0, c_0), (v_1, s_1, a_1, c_1) \cdots \in (V \times S^k \times A^k \times C^k)^\omega$. We call such an infinite sequence a *play* and let *Plays* denote the set of all plays.

Note that games as described above are *partial-* (aka. *imperfect*) information games: the players do not directly observe the current state, the actions selected by other players nor the rewards obtained (not even their own). All they get to see is the signals sent to them. We call a game *perfect-information* if, at every stage $i > 0$, every player receives as a signal the full action vector a_{i-1}, reward vector c_{i-1} and the current state v_i. A game is *turn-based* (as opposed to *concurrent*) if in any stage, only one player's action influences the next state and signals. That is, for every vertex $v \in V$ there is some player j so that $\Delta(v, a) = \Delta(v, a')$ whenever $a[j] = a'[j]$.

Plays and Strategies. A *strategy* is a function $\sigma : S^* \to \mathbb{D}(A)$, that determines, for every sequence of signals, a probability distribution over the action set. It is based on a set M of memory modes if it can be described by an initial memory mode $m_0 \in M$ and a pair of functions $\sigma_{act} : M \times S \to \mathbb{D}(A)$ and $\sigma_{up} : M \times S \to \mathbb{D}(M)$ that select actions and update the memory mode, respectively. That is, for any signal sequence, $s = s_0 s_1 s_2 \dots s_j \in S^*$, $\sigma(s) = \sigma_{act}(m_j, s_j)$ and $m_{j+1} \stackrel{\text{def}}{=} \sigma_{up}(m_j, s_j)$.

A strategy is called *finite-memory* if it is based on a finite memory set M and *memoryless* if it is based on a singleton set M. A strategy is *discrete* (also *pure*) if all its choices are Dirac. In perfect-information games, we call a strategy *stationary* if its choices only depend on the current state (and not previous actions or rewards).

Strategy profiles, Probability measures. A *strategy profile* is a family $\sigma = (\sigma[j])_{j \leq k}$ of strategies, one for each player. Together with an initial state v_0 and signal s_0, it uniquely induces a probability

measure $\mathbb{P}^{\sigma}_{v_0,s_0} \in \mathbb{D}(Plays)$ over infinite plays (see [3] for details). We will write $\mathbb{E}^{\sigma}_{v_0,s_0}[X]$ for the derived expected value of a random variable X and drop indices when clear from the context.

Objectives. Each player $j \leq k$ has an objective, a measurable function $O_j : Plays \to \mathbb{R}$ and selects their strategy to maximise the expectation of this objective, against any possible opponent strategies. A game is *zero-sum* if $\sum_{j\leq k} O_j(\rho) = 0$ for every $\rho \in Plays$.

In this paper, we consider mean-payoff objectives as well as the special case of reachability objectives. A *mean-payoff* objective (for player $j \leq k$) assigns play $\rho = (v_0, s_0, a_0, c_0), (v_1, s_1, a_1, c_1) \ldots$ the value

$$\underline{MP}(\rho) = \liminf_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N} c_i[j].$$

The *reachability* objective for a target set $T \subseteq V$ of states assigns play ρ as above the value 1 if $v_i \in T$ for some round $i \geq 0$ and 0 otherwise.

Optimality criteria. The player in a 1-player game G plays to maximise the expectation of the given objective O. Here, the *value* of the game is $\mathtt{val}(G) \stackrel{\text{def}}{=} \sup_{\sigma\in\Sigma} \mathbb{E}^{\sigma}_{v_0,s_0}[O]$ and for $\varepsilon \geq 0$, a strategy σ is called *ε-optimal* if $\mathtt{val}(G) - \mathbb{E}^{\sigma}_{v_0,s_0}[O] \leq \varepsilon$.

For 2-player zero-sum games G, let Σ and Π denote the set of strategies for players 1 and 2, respectively. Then the *lower value* and *upper value* are defined as $\overline{\mathtt{val}}(G) \stackrel{\text{def}}{=} \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \mathbb{E}^{\sigma,\pi}_{v_0,s_0}[O]$ and $\underline{\mathtt{val}}(G) \stackrel{\text{def}}{=} \inf_{\pi\in\Pi} \sup_{\sigma\in\Sigma} \mathbb{E}^{\sigma,\pi}_{v_0,s_0}[O]$. By definition, it holds that $\overline{\mathtt{val}}(G) \leq \underline{\mathtt{val}}(G)$. If the two are equal then this quantity is called the *value* of the game, denoted by $\mathtt{val}(G)$. For $\varepsilon \geq 0$, a strategy $\sigma \in \Sigma$ is called *ε-optimal* if $\mathtt{val}(G) - \inf_{\pi\in\Pi} \mathbb{E}^{\sigma,\pi}_{v_0,s_0}[O] \leq \varepsilon$ (and similar for player 2 strategies π).

For general k-player games G and any $\varepsilon \geq 0$, an *ε-Nash equilibrium* (NE) is a strategy profile $(\sigma_j)_{j\leq k}$ so that any unilateral deviation by a player j increases their expected outcome by at most ε. That is, for any profile σ' where $\sigma'[i] = \sigma[i]$ for all $i \neq j$, it holds that $\mathbb{E}^{\sigma'}[O_j] \leq \mathbb{E}^{\sigma}[O_j] + \varepsilon$.

Complexity classes. Function problems are given by a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ where for some polynomial p, every $(x, y) \in R$ implies $|y| \leq p(|x|)$. For a class $\mathcal{X}$ of decision problems, R is in the corresponding function class $\mathsf{F}\mathcal{X}$ if and only if the language $\{(x, y) \mid R(x, y)\}$ is in $\mathcal{X}$, meaning the question if $R(x, y)$ is true for given x and y is decidable in $\mathcal{X}$. This corresponds to the search problem where for given x, one can either stop and output y so that $R(x, y)$, or correctly state that no such y exists. In particular, we are concerned with the classes $\mathsf{FNP}^{\mathsf{NP}}$, at level 2 of the polynomial hierarchy.

Floating-point Representations of Distributions. For any number $u \in \mathbb{N}$, let $\mathbb{Q}(u)$ denote the set of non-negative dyadic rationals with a u-bit mantissa

$$\mathbb{Q}(u) \stackrel{\text{def}}{=} \{0\} \cup \{x2^{-i} \mid x \in \{2^{u-1}, 2^{u-} + 1, \ldots, 2^u - 1\}, i \in \mathbb{Z}\}.$$

The u-bit floating point representation of $x2^{-i}$ is $\langle 1^u, [x]_2, [i]_2\rangle$, where $[\cdot]_2$ is the binary expansion. The floating point representation of 0 is $\langle 1^u, 0\rangle$. The *relative distance* between two non-negative real numbers x and x' as $dist(x, x') = \frac{\max(x,x')}{\min(x,x')} - 1$ with the convention that $0/0 = 1$ and $c/0 = +\infty$ for any $c > 0$. Two non-negative reals x and x' are *(u, i)-close* if $dist(x, x') \leq \left(\frac{1}{1-2^{-u+1}}\right)^i - 1$. Let $\mathbb{P}(u)$ denote the set of finite probability distributions $(p_1, p_2, \ldots, p_n)$ so that there exist numbers $(p'_1, p'_2, \ldots, p'_n) \in \mathbb{Q}(u)$ with $p_i = p'_i / \sum_j p'_j$ for all $i = 1, \ldots, n$ and so that $\sum_j p'_j$ is (u, n)-close to 1. We call the primed vector the (u-bit) *approximately normalised representation* of the unprimed distribution.

3 OVERVIEW OF OUR RESULTS

We consider the problem of approximating ε-Nash equilibria and zero-sum values with respect to bounded-memory strategies.

Definition 3.1. Consider some k-player game with objectives $(O_j)_{j\leq k}$ and a set $\mathcal{S}$ of admissible strategy profiles. Let $\varepsilon \geq 0$.

An ε-Nash equilibrium *with respect to* $\mathcal{S}$ is a strategy profile $(\sigma_j)_{j\leq k} \in \mathcal{S}$ so that any unilateral deviation by a player j that yields another profile in $\mathcal{S}$ can at most increase their expected outcome by ε. That is, for any profile $\sigma' \in \mathcal{S}$ where $\sigma'[i] = \sigma[i]$ for all $i \neq j$, it holds that $\mathbb{E}^{\sigma'}[O_j] \leq \mathbb{E}^{\sigma}[O_j] + \varepsilon$.

In the 2-player zero-sum case, we say a strategy σ for player 1 *ε-achieves* value $v \in \mathbb{R}$ with respect to $\mathcal{S}$ if it guarantees expected outcome of at least $v - \varepsilon$ against all admissible opponent strategies. That is, if $\inf_{\pi \mid (\sigma,\pi)\in\mathcal{S}} \mathbb{E}_{\sigma,\pi}[O_1] \geq v - \varepsilon$.

For instance, an ε-Nash equilibrium with respect to finite-memory strategies, is a profile consisting of finite-memory strategies, so that no player can, by switching to another finite-memory strategy, improve their expected payout by more than ε.

We are interested in approximating such equilibria and zero-sum strategies with respect to finite memory strategies with bounded memory. Formally, we are given a k-player partial-information game G, some $\varepsilon \in [0, 1]$ and a vector $\mathbf{b} \in \mathbb{N}^k$ of memory bounds, one for each player $i \leq k$. Write $\mathcal{B}_i(\mathbf{b})$ for the set of player i strategies that are based on a set M of size $|M| \leq \mathbf{b}[i]$ and let $\mathcal{B}(\mathbf{b})$ be all strategy profiles wherein each player i uses a strategy in $\mathcal{B}_i(\mathbf{b}[i])$. For multiplayer games ($k \geq 2$) we ask if there exists an ε-Nash equilibrium with respect to (wrt) $\mathcal{B}(\mathbf{b})$ and if so, we want to compute one. For 2-player zero-sum games, we ask if there exists a strategy that ε-achieves a given value v with respect to (wrt) $\mathcal{B}(\mathbf{b})$.

Throughout, we assume that the number k of players is fixed; The inputs ε, v and rewards are given in binary and the bounds $\mathbf{b}$ in unary encoding[1]. The probability distributions dictated by the transition function Δ are assumed to be given in u-bit approximately normalised floating point notation.

Theorem 3.2. *For every fixed $k \geq 1$ the following is in* PSPACE.

1. *Given a k-player game, bounds $\mathbf{b} \in \mathbb{N}^k$ and $\varepsilon \geq 0$, check whether an ε-Nash equilibrium exists with respect to $\mathcal{B}(\mathbf{b})$.*
2. *Given a 2-player zero-sum game, bounds $\mathbf{b} \in \mathbb{N}^2$, a number $v \in \mathbb{R}$ and $\varepsilon \geq 0$, check whether there exists a player 1 strategy that ε-achieves value v with respect to $\mathcal{B}(\mathbf{b})$.*

Our proof is by an effective polynomial reduction of the problem to the satisfiability problem of FO($\mathbb{R}$) formula so that the size of the constructed formula is only polynomial in the input, has fixed alternation depth of 2, and solutions dictate values and witnessing

[1] This assumption is natural as we want to compute representations of strategies, in particular, which mixed action to play for any combination of game and memory state.

strategies/equilibria. At its core, our encoding relies on a state reduction technique for finite Markov chains, that results from fixing a strategy profile. This requires only polynomially many collapses and can therefore be "implemented" as a small system of polynomial equations.

We show that equilibria, zero-sum values and witnessing strategies can be approximated, up to an exponential additive error, at level 2 of the polynomial hierarchy.

Theorem 3.3. *For every fixed $k \geq 1$ the following are in* $\mathsf{FNP}^{\mathsf{NP}}$.

(1) Given a k-player game, bounds $b \in \mathbb{N}^k$ and $\varepsilon \in (0, 1/2)$, find an ε-Nash equilibrium with respect to $\mathcal{B}(\mathbf{b})$ (if for some $\varepsilon' \in [0, \varepsilon/3)$, an ε'-Nash equilibrium with respect to $\mathcal{B}(\mathbf{b})$ exists);

(2) Given a 2-player zero-sum game, bounds $b \in \mathbb{N}^k$, $\varepsilon \in (0, 1/2)$ and $v \in \mathbb{R}$, find a strategy that ε-achieves v with respect to $\mathcal{B}(\mathbf{b})$ (if for some $\varepsilon' \in [0, \varepsilon/3)$, some strategy ε'-achieves v with respect to $\mathcal{B}(\mathbf{b})$).

Using the second point and binary search one can ε-approximate the lower and upper values wrt $\mathcal{B}(\mathbf{b})$ in 2-player zero-sum games. These complexity upper bounds above directly apply to many classes of games that admit memoryless ε-optimal strategies ($\mathcal{B}(\mathbf{1})$).

In addition to the approximation algorithms, we provide an NP- and coNP-lower bound for approximating zero-sum values for partial-information games. This already holds for a very restricted class of games, as follows. Let us call a game *simple* if it is a 1-player reachability game so that the underlying transition graph is acyclic (except for target states); the motion from the initial state v_0 is uniformly at random over a set of successor states; and for all actions a and states $v \neq v_0$ the distributions $\Delta(v, a)$ are Dirac.

We show that it is hard to check if the value achieved by memoryless strategies exceeds a given threshold. We state this in the form of a gap problem (cf. [23]).

Theorem 3.4. *The following decision problem is* NP*-hard. Given $0 \leq l < u \leq 1$ in unary encoding and a simple game in which the zero-sum value is not in (l, u) and in which there exists an optimal memoryless deterministic strategy. Is the value greater than or equal to u?*

This lower bound directly applies to threshold problems regarding the values of zero-sum games, approximating zero-sum values and witnessing strategies, and to approximating ε-equilibria in multiplayer games, wrt memoryless strategies ($\mathcal{B}(\mathbf{1})$).

4 LOWER BOUND

We prove Theorem 3.4 by reduction from the Boolean satisfiability problem: an instance of 3SAT with n clauses and m variables variables $X_1, X_2, \ldots, X_m$ is a conjunction

$$\varphi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} L_{i,j}$$

where each conjunct is called a clause and each $L_{i,j}$ is either a positive literal (a variable in $\{X_k \mid k \leq m\}$) or a negative literal (the negation of a positive literal). W.l.o.g., assume that no variable appears both positively and negatively in any clause. An *interpretation* is a mapping $\nu : \{1, \ldots, m\} \to \{true, false\}$ assigning a truth value to each variable. The (NP-hard) decision problem is if there exists an interpretation ν under which the given conjunction is true, meaning that for every clause i there is some j such that either $L_{i,j} = X_k$ and $\nu(k) = true$ or $L_{i,j} = \neg X_k$ and $\nu(k) = false$. Analogously, it is coNP-hard to check if, for a given 3SAT instance, no interpretation satisfies the given conjunction.

We proceed to describe a game G_φ for a given instance φ. This game is set up so that in the first round, independently of the player's action, a random clause is selected to be checked. In the following m rounds, the player's actions determine an interpretation, one variable per round and without knowing which clause is being checked.

Formally, for every $0 \leq j \leq m+1$ the game has a state (w, j) as well as states (i, j) for all $0 \leq i \leq n$. The available actions are $A = \{true, false\}$. The game starts in $v_0 = (0, 0)$ and in the first round, under both actions, moves uniformly at random into a state in $\{(i, 1) \mid 0 \leq i \leq n\}$, thereby selecting the clause i to be checked.

In round $0 < j \leq m$, the player picks an action $true$ or $false$ to indicate that variable X_j has that value in his chosen interpretation. From state (w, j), the pebble moves surely to $(w, j+1)$, regardless of this choice. Otherwise, from state $(i, j) \neq (w, j)$, there are three cases depending on whether clause C_i is satisfied by setting the value of X_j as indicated by the player's action (cf. Figure 1).

(a) If C_i contains X_j positively, then upon action $true$ the pebble surely moves to $(w, j+1)$ and upon action $false$, it surely moves to $(i, j+1)$.

(b) If C_i contains X_j negatively, then the roles are reversed: upon action $false$ the pebble surely moves to $(w, j+1)$ and upon action $true$, it surely moves to $(i, j+1)$.

(c) If C_i does not contain X_j, then upon both actions the pebble surely to $(i, j+1)$.

In every round from a state $(*, j)$, the player receives signal j. This ends the description of game G_φ.

The player only knows the current round number j up to $m+1$, but not in which state the pebble resides, i.e., which clause is being checked and whether it is already satisfied. By his actions in rounds $1, \ldots, m$, he picks a distribution over the possible variable interpretations.

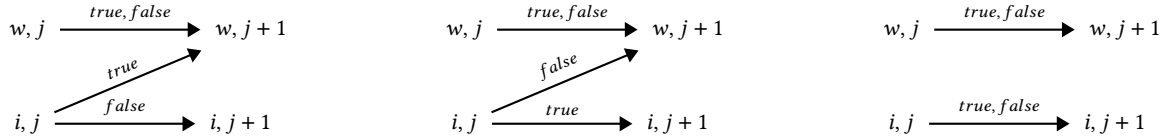
Let us write $W \stackrel{\text{def}}{=} \mathrm{Reach}(\{(w, m+1)\})$ for the set of plays that reach state $(w, m+1)$. The following formalises the key property of the game G_φ.

Lemma 4.1. *Let ν be an interpretation that satisfies b out of n clauses and let σ be a strategy that plays action $\nu(X_j)$ in round $0 < j \leq m$. Then $\mathbb{P}^{\sigma}_{v_0,0}(W) = b/n$.*

Proof. In the first round, nature picks uniformly at random a state $(i, 1)$ for some $0 < i \leq n$. Then in exactly b out of n many cases (the ones where clause C_i is satisfied by the assignment), the play leads to $(w, m+1)$. □

Proof of Theorem 3.4. To show the NP-hardness, we reduce from 3SAT. Given an instance φ with n clauses and m variables, let c be the maximal number of clauses satisfied by any interpretation.

Let G be the game G_φ where the player receives reward 1 whenever the pebble is in state $(w, m+1)$ and 0 otherwise. This game is *simple*; in particular, it is a 1-player reachability game with target $(w, m+1)$, encoded as a mean-payoff game.



(a) C_i contains X_j. (b) C_i contains $\neg X_j$. (c) C_i contains neither X_j nor $\neg X_j$

Figure 1: Stages $0 < j \leq m$ in game G_φ. In step $j \leq m$ the player receives signal j and the pebble is moved out of some state $(*, j)$.

We show that the value in G is 1 if φ is satisfiable and at most $1 - 1/n$ otherwise. Suppose first that φ is satisfiable. Then there exists an interpretation that satisfies $c = n$ clauses and the value is 1, by Lemma 4.1. Otherwise, if φ is not satisfiable, every interpretation satisfies at most c clauses and thus witnesses at most value c/n. As the history of signals is the same for all infinite plays, every (arbitrary) strategy σ defines a distribution over all possible variable interpretations. The value it witnesses is the linear combination of the values witnessed by discrete strategies, and so also at most $c/n \leq 1 - 1/n$.

Let $l \stackrel{\text{def}}{=} 1 - 1/n$ and $u \stackrel{\text{def}}{=} 1$. By the argument above, the value of the game is not in (l, u) and $\geq u$ if, and only if, φ is satisfiable. □

By the argument above, it is coNP-hard to check if the value of a simple game is less than or equal to a threshold l given in unary. Similarly, one can introduce a passive second player that receives inverted rewards ($-r$ whenever player 1 gets r). This shows that it is coNP-hard to check if the value of a 2-player zero-sum safety game exceeds a threshold u.

The games constructed above all admit memoryless optimal strategies. Therefore, approximating the value wrt memoryless ($\mathcal{B}(b = 1)$) strategies is at best FNP, even for simple games.

5 MEAN-PAYOFF VALUES FOR MARKOV CHAINS

We demonstrate how to iteratively collapse Markov chains with the aim of computing the expected mean-payoff value of a fixed initial state. The underlying idea is to simultaneously summarise the expected reward and duration (number of steps) of paths between two nodes and adjust these summaries consistently during collapses. Once an irreducible chain is produced it is trivial to read off the mean-payoff values.

A Markov chain is a 1-player game with only a single, unique action $a \in A$ for the player in every state (the player is just there to collect the rewards and cannot influence the outcome). To simplify notations, we will assume that a Markov chain M has states $V = \{1, \ldots, n\}$ and for every edge from states i to j, let $p(i, j)$ and $r(i, j) \in \mathbb{R}$ be the associated probability and reward respectively. Each edge from i to j also has a duration denoted by $d(i, j)$. We define the *mean-payoff ratio* of an infinite play $\rho = s_0 s_1 \ldots$ as the limit of total reward by the total duration of expanding prefixes:

$$MPR(\rho) \stackrel{\text{def}}{=} \liminf_{N \to \infty} \frac{\sum_{i=0}^{N-1} r(s_i, s_{i+1})}{\sum_{i=0}^{N-1} d(s_i, s_{i+1})} \tag{1}$$

Notice that this coincides with the usual mean-payoff $\underline{MP}(\rho)$ if all edges have duration 1.

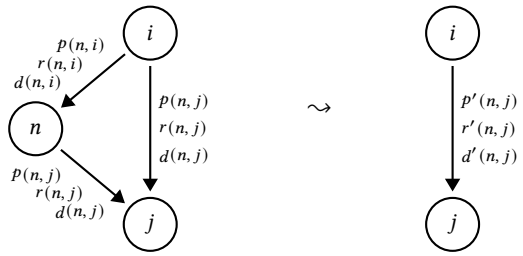


Figure 2: Elimination of state n: every length-two path via state n in M (on the left) is removed and the corresponding direct edge re-weighted in M' (right). The expected reward and duration of going from i to j remains the same.

Let's recall a few notions about Markov chains. Write $i \to j$ to denote that there is a path from state i to j with non-zero probability. We say i and j *communicate* if both $i \to j$ and $j \to i$. A set $S \subseteq V$ of states is communicating if all its elements are pairwise communicating. S is closed if $i \to j$ and $i \in S$ implies $j \in S$. Closed communicating sets (CCSs) are also called bottom strongly connected components. A state i is *recurrent* if it belongs to a CCS and *transient* otherwise. It is *absorbing* if $p(i, i) = 1$. That is, an absorbing state describes its own (singleton) CSS.

We introduce two operations on Markov chains that will preserve the expected mean-payoff ratio $\mathbb{E}_s[MPR]$ from any state s. Proofs of their correctness (Lemmas 5.2 and 5.4) are in Appendix A.

Definition 5.1 (State Elimination). Let $M = (V, p)$ be a Markov chain with states $\{1, \ldots, n\}$ and probability matrix p so that $p(n, n) = 0$ (state n does not have a self-loop). The Markov chain $M' = (V', p')$ results from M by eliminating state n if it has states $V' = V \setminus \{n\}$ and, for every two states $i, j \neq n$, the values of probabilities, rewards, and durations are updated as follows (see Figure 2).

$$p'(i, j) \stackrel{\text{def}}{=} p(i, j) + p(i, n) \cdot p(n, j)$$

$$r'(i, j) \stackrel{\text{def}}{=} p(i, j)\, r(i, j) + (p(i, n) \cdot p(n, j))(r(i, n) + r(n, j))$$

$$d'(i, j) \stackrel{\text{def}}{=} p(i, j)\, d(i, j) + (p(i, n) \cdot p(n, j))(d(i, n) + d(n, j))$$

Lemma 5.2. *Suppose M' results from M by eliminating state n. Then $\mathbb{E}_s^M[MPR] = \mathbb{E}_s^{M'}[MPR]$ for every state $s \neq n$.*

Definition 5.3 (Loop Elimination). Let $M = (V, p)$ be a Markov chain with states $\{1, \ldots, n\}$ and probability matrix p so that $0 < p(n, n) < 1$ (state n is not absorbing and has a self-loop). Eliminating loop n results in a Markov chain $M' = (V', p')$ with $V' = V$

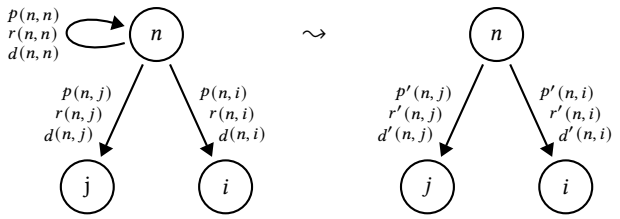


Figure 3: Elimination of loop n in M (left). In M' (right), the corresponding edge has probability 0 and all other edges from n are re-weighted to match the expected duration and expected reward of paths that start by iterating the loop.

where for any edge from state $i \neq n$ to j, the probability, reward, and duration are as in M. For edges going out of state n they are $p'(n,n) = r'(n,n) = d'(n,n) = 0$ and for all $j \neq n$,

$$p'(n,j) \stackrel{\text{def}}{=} \frac{p(n,j)}{1-p(n,n)}$$

$$r'(n,j) \stackrel{\text{def}}{=} \frac{1}{1-p(n,n)}\, r(n,n) + \frac{p(n,j)}{1-p(n,n)}\, r(n,j)$$

$$d'(n,j) \stackrel{\text{def}}{=} \frac{1}{1-p(n,n)}\, d(n,n) + \frac{p(n,j)}{1-p(n,n)}\, d(n,j).$$

The intuition behind this definition is that iterating the loop $n \to n$ and then going to $j \neq n$ is replaced by a direct step that preserves the probability of ending in that particular state j, as well as the expected sum of rewards and durations on the way. Recall that the expected number of times the loop is used is $1/1-p(n,n)$.

Lemma 5.4. *Suppose M' results from M by eliminating loop n. Then $\mathbb{E}_s^M[MPR] = \mathbb{E}_s^{M'}[MPR]$ for every state $s \leq n$.*

The two elimination operations can be used to determine mean-payoff ratio values in finite Markov chains. Indeed, suppose we want to determine $\mathbb{E}_s^M[MPR]$. Exhaustively eliminating all loops and states except s results in a Markov chain M' where every original CCS is collapsed into a single absorbing state. By Lemmas 5.2 and 5.4, the expected mean-payoff ratio of state s is the same in M and M'. Now, if M' contains only a single absorbing state s, then trivially $\mathbb{E}_s^{M'}[MPR] = r'(s,s)/d'(s,s)$. Otherwise, if s is the only transient state in M' then $\mathbb{E}_s^{M'}[MPR] = \sum_{i \neq s} p'(s,i)\,(r'(i,i)/d'(i,i))$.

Notice that the number of elimination steps is at most $2|V|$. We will implement this procedure symbolically in FO($\mathbb{R}$) in the next section, and approximately, in Section 7.4, to prove our main results.

6 EXPRESSIBILITY IN FO($\mathbb{R}$)

We fix a k-player game G, bounds $\mathbf{b} \in \mathbb{N}^k$ and let

$$N \stackrel{\text{def}}{=} |V| \cdot |S|^k \cdot \prod_{i=1}^{k} \mathbf{b}[i].$$

We show how to express ε-Nash equilibria wrt $\mathcal{B}(\mathbf{b})$ (in the multiplayer case) and strategies that ε-achieve some value v wrt $\mathcal{B}(\mathbf{b})$ (in the 2-player zero-sum case) in FO($\mathbb{R}$). The steps to do this are as follows. We will elaborate on each step in subsequent paragraphs.

(1) We existentially guess a strategy profile (in the multiplayer case), or a strategy for player 1 (in the 2-player, zero-sum case), such that the probability distributions used in the strategies are encoded in the variables.
(2) For any player i, we have a formula that depends on variables encoding a strategy profile, and the formula encodes the outcome for the player i in the Markov chain induced by the strategy profile.
(3) We encode that the strategies guessed in step 1 satisfy the desired criteria.

Step 1: Guessing strategies and strategy profiles. Encoding a strategy in $\mathcal{B}_i(\mathbf{b})$ for any player i (and thus strategy profiles in $\mathcal{B}(\mathbf{b})$), in FO($\mathbb{R}$) is straightforward. Indeed, a strategy in $\mathcal{B}_i(\mathbf{b})$ consists of an action function $\sigma_{act} : M \times S \to \mathbb{D}(A)$ and a memory update function $\sigma_{up} : M \times S \to \mathbb{D}(M)$. For the action function, we have for each signal, each memory state and each of the player's possible actions under that signal, a variable. Specifically, if the actions with signal s and memory m are $a_1, \ldots, a_{|A|}$, we have variables $q_1^{s,m,i}, \ldots, q_{|A|}^{s,m,i}$. To express that they should form a probability distribution, we have inequalities $0 \leq q_j^{s,m,i} \leq 1$, for each $j \in \{1, \ldots, |A|\}$ and $\sum_{j=1}^{|A|} q_j^{s,m,i} = 1$.

Similarly, for the memory update function σ_{up}, we just, for each signal s, each memory m, and each action vector a in that state have variables $b_1^{s,m,a,i}, \ldots, b_{\mathbf{b}[i]}^{s,m,a,i}$. To express that they should form a probability distribution, we have inequalities $0 \leq b_j^{s,m,a,i} \leq 1$ for each $j \in \{1, \ldots, \mathbf{b}[i]\}$ and $\sum_{j=1}^{\mathbf{b}[i]} b_j^{s,m,a,i} = 1$. This gives us the following lemma.

Lemma 6.1. *For any k-player game G and $i \leq k$, one can construct a FO($\mathbb{R}$) formula $Strat_i(\mathbf{x_i})$, with $|\mathbf{x_i}| = (|S| \cdot \mathbf{b}[i])(|A| + \mathbf{b}[i])$ variables, such that there is a one-to-one correspondence between satisfying assignments of $\mathbf{x_i}$ and strategies in $\mathcal{B}_i(\mathbf{b})$ for player i.*

Step 2: Evaluating the induced Markov chain. Once the strategy profile is fixed, we obtain an induced Markov chain. The following lemma shows that there is an FO($\mathbb{R}$) formula that uses variables to identify a strategy profile and encodes the value obtained by the strategy profile by encoding the loop and state elimination procedure from Section 5. The proof, showing how this formula is constructed is in Appendix B. In essence, the loop/state elimination procedures are iteratively implemented in FO(R) by introducing lots (but polynomially many) existentially quantified variables to represent intermediate Markov chains in the reductions.

Lemma 6.2. *For any k-player game G, and a player i, one can construct a formula $Val_i(\mathbf{x}_1, \ldots, \mathbf{x}_k, y)$, such that $\mathbf{x}_j$ encodes strategy σ_j of a strategy profile $\sigma \in \mathcal{B}(\mathbf{b})$, and the formula uses a set of Boolean combination of $100N^3 + 40N$ polynomials with at most $8N^3 + 3N$ variables, each of degree at most $2k$, and coefficients coming from G and y encode the outcome of σ for player i.*

Step 3: Finding ε-achieving strategy/ε-Nash equilibria. The last step requires unfolding the definition of ε-achieving strategy/ε-Nash equilibria in a formula. We first do this for the 2-player zero-sum case where we are interested in strategies that ε-achieves a given value v, for $\varepsilon \geq 0$. The formula guesses the ε-achieving strategy, and checks for each of the opponent's strategies, that the

outcome is $\geq v - \varepsilon$.

$$\exists \mathbf{x_1} \forall \mathbf{x_2} \forall y. \quad \mathrm{Strat}_1(\mathbf{x_1}) \wedge \mathrm{Strat}_2(\mathbf{x_2}) \wedge \mathrm{Val}_1(\mathbf{x_1}, \mathbf{x_2}, y) \implies (y \geq v - \varepsilon)$$

Note that one can use existentially quantified auxiliary variables to implement repeated squaring and thereby encode the (binary) numbers v and ε succinctly.

For the case of ε-Nash equilibria wrt $\mathcal{B}(\mathbf{b})$, we guess the strategy profile σ and check against all possible deviations by a single player.

$$\exists \mathbf{x_1}, \ldots, \mathbf{x_k}, y_1, \ldots, y_k \forall \mathbf{x'_1}, \ldots, \mathbf{x'_k}, y'.$$
$$\left(\bigwedge_{i=1}^{k} \mathrm{Strat}_i(\mathbf{x_i}) \right) \wedge \left(\bigwedge_{i=1}^{k} \mathrm{Val}_i(\mathbf{x_1}, \ldots, \mathbf{x_k}, y_i) \right) \wedge$$
$$\bigwedge_{j=1}^{k} \left(\mathrm{Strat}_j(\mathbf{x'_j}) \implies (\mathrm{Val}_j(\mathbf{x_1}, \ldots, \mathbf{x'_j}, \ldots, \mathbf{x_k}, y') \wedge y_j + \varepsilon \geq y') \right)$$

Lemma 6.3. *Given a k-player game, and $v, \varepsilon \geq 0$, one can construct an* FO($\mathbb{R}$) *formula that encodes a strategy that ε-achieves v/ is a ε-Nash equilibrium. Furthermore, the formula uses at most $(k+1) + k(|S| \prod_{i=1}^{k} \mathbf{b}[i])(|A| + \sum_{i=1}^{k} \mathbf{b}[i]) + 8N^3 + 3N$ variables, and at most $(100N^3 + 40N)2k$ polynomials.*

The terms in the number of variables correspond to variables for guessing values, strategies and the variables used to describe the values in Lemma 6.2.

Proof of Theorem 3.2. From Lemma 6.3, we can express strategies encoding an ε-Nash equilibrium using an FO($\mathbb{R}$) formula. Note that this uses 2 alternations of quantifiers. Checking if a formula with bounded quantifier alternation is in the first-order theory of the reals can be done in PSPACE as shown by Basu et al.[2, Remark 13.10]. □

The following will be used later applied to the formulae provided by Lemma 6.3. For the proof, we assume familiarity with FO($\mathbb{R}$), the first order theory of the reals [2].

Lemma 6.4. *Consider a satisfiable* FO($\mathbb{R}$) *expression,*

$$\exists x_1^1, \ldots, x_{n_1}^1 \forall x_1^2, \ldots, x_{n_2}^2 \exists \ldots x_1^k \ldots, x_{n_k}^k . \varphi(x_1^1, \ldots, x_{n_1}^1, x_1^2 \ldots, x_{n_k}^k)$$

where φ consists of boolean combinations over s polynomial inequalities, where each polynomial is of degree at most d, has integer coefficients and the bit-size of each coefficient is at most τ. Then there exists $x'_1, \ldots, x'_{n_1}$, such that,

$$\forall x_1^2, \ldots, x_{n_2}^2 \exists \ldots \exists x_1^k \ldots, x_{n_k}^k . \varphi(x'_1, \ldots, x'_{n_1}, x_1^2 \ldots, x_{n_k}^k)$$

is satisfiable and such that for each i, either $2^{-\tau(2d)^{\prod_{i=1}^{k} O(n_i)}} \leq |x'_i| \leq 2^{\tau(2d)^{\prod_{i=1}^{k} O(n_i)}}$ or $x'_i = 0$.

Proof. We apply [2, Thm 14.16] to

$$\phi(x_1^1, \ldots, x_{n_1}^1) \stackrel{\text{def}}{=} \forall x_1^2, \ldots, x_{n_2}^2 \; \exists \ldots x_1^k \ldots, x_{n_k}^k : \varphi(x_1^1, \ldots, x_{n_1}^1, x_1^2 \ldots, x_{n_k}^k)$$

which gives us a quantifier-free expression of ϕ. This expression of ϕ uses polynomials of degree at most $d' \stackrel{\text{def}}{=} d^{\prod_{i=2}^{k} O(n_i)}$ with integer coefficients, each of bit-size at most $\tau' \stackrel{\text{def}}{=} \tau d^{\prod_{i=1}^{k} O(n_i)}$. We then apply [2, Thm 13.10] to ϕ, giving us a "univariate representation" of each of the numbers we will use for x'_i (the theorem itself only states that we get approximations, but the complexity analysis of the algorithm states that it is equally fast to get exact solutions, using some additional steps). For each i, this univariate representation for x_i consists of a fraction of two polynomials $p_i(x)/q(x)$ and an additional polynomial f (q and f are coprime). The number x_i is then $p_i(t)/q(t)$ where t is a root of f. These polynomials each is of degree at most $d'' \stackrel{\text{def}}{=} (2d'+6)^{n_1} \leq (2d)^{\prod_{i=1}^{k} O(n_i)}$ (we get to ignore the 6 because of the O's. We can't ignore the 2 in case d was 1) with integer coefficients of bit-size at most $\tau'' \stackrel{\text{def}}{=} \tau d^{\prod_{i=1}^{k} O(n_i)}$. For each i, we apply [2, Lem. 15] to f, p_i and q, giving us a square-free univariate polynomial p'_i such that $p'_i(x_i) = 0$. This polynomial has degree at most $d''' \stackrel{\text{def}}{=} 2d'' = (2d)^{\prod_{i=1}^{k} O(n_i)}$ and the bitsize representation of the integer coefficients is at most $\tau''' \stackrel{\text{def}}{=} 2d''\tau'' + 7d'' \log d'' = \tau(2d)^{\prod_{i=1}^{k} O(n_i)}$.

For any univariate polynomial $p(x) = a_n x^{n-1} + a_{n-1}x^{n-2} + \cdots + a_1 x + a_0$, then $r \neq 0$ is a root iff $1/r \neq 0$ is a root of $a_n 1/(x^{n-1}) + a_{n-1}(1/x^{n-2}) + \cdots + a_1 1/x + a_0$ or equally $p'(x) \stackrel{\text{def}}{=} a_n + a_{n-1}x + \cdots + a_1 x^{n-2} + a_0 x^{n-1}$. Therefore, if U is an upper bound on the absolute largest root in p, then $1/U$ is a lower bound for how small the absolute smallest non-zero root can be in p'. We can bound $U \leq 1 + \max\left(\left| \frac{a_{n-1}}{a_n} \right|, \left| \frac{a_{n-2}}{a_n} \right|, \ldots \left| \frac{a_0}{a_n} \right| \right)$ [11] and so get that $2^{-\tau'''} \leq |x'_i| \leq 2^{\tau'''}$, because the largest integer represented with x bits is $2^x - 1$. □

7 APPROXIMATION ALGORITHMS

In this section, we present the proof of Theorem 3.3. Our approach is to repeat the steps leading to FO($\mathbb{R}$) representations in the previous section but use floating point representations and associated lossy arithmetic. To do this, we need two ingredients: (1) we need to guess candidate profiles in polynomial space; and (2) we need to compute approximations of the mean-payoff values in the induced Markov chains in polynomial time.

We focus first on the existence of small candidate profiles.

7.1 Floating point representations

Definition 7.1. A strategy profile $\sigma = (\sigma[j])_{j \leq k} \in \mathcal{B}(\mathbf{b})$ is *u-bit representable* if all its distributions (selecting mixed actions and memory updates for all players, memory modes and signals) are in $\mathbb{P}(u)$. The profile σ is *represented* in u-bits if it is u-bit representable and all exponents are written in binary using u many bits.

Let

$$p(\sigma, \mathbf{m}, s, \mathbf{m'}, \mathbf{a}) \stackrel{\text{def}}{=} \prod_{j=1}^{k} \sigma[j]_{act}(\mathbf{m}[j], s[j])(\mathbf{a}[j]) \cdot \sigma[j]_{up}(\mathbf{m}[j], s[j])(m'[j])$$

denote the probability of that starting with memory states $\mathbf{m}$ and having previously received signal vector s, the players jointly pick the action profile $\mathbf{a}$, and update the memory vector to $\mathbf{m'}$.

For any two profiles $\sigma, \sigma' \in \mathcal{B}(\mathbf{b})$, the *distance* between the strategy profiles is

$$dist(\sigma, \sigma') \stackrel{\text{def}}{=} \max_{\mathbf{m}, s, \mathbf{m'}, \mathbf{a}} dist(p(\sigma, \mathbf{m}, s, \mathbf{m'a}), p(\sigma', \mathbf{m}, s, \mathbf{m'}, \mathbf{a})).$$

7.2 Polynomial witnesses

We now show the existence of suitably small profiles to witness approximate equilibria and zero-sum values. Our claim that small profiles exist is stated in Lemma 7.5. This relies on the following two lemmas, proofs of which can be found in the appendix.

For the remainder of the section, fix a k-player game G with n states and where $C \in \mathbb{N}$ is the largest absolute value of any reward. Also, fix some vector $\mathbf{b} \in \mathbb{N}^k$ of memory bounds. Let $N \stackrel{\text{def}}{=} n \cdot |S|^k \cdot \prod_j \mathbf{b}[j]$.

The first lemma states that if there are witnesses at all, then there are also ones where all probabilities are at most double exponentially small.

Lemma 7.2 (Lower-bounded probabilities). *There is $u_0 \in \mathbb{N}$, polynomial in the size of the game and ε, so that the following is true for all $u \geq u_0$.*

If there exists a profile $\sigma \in \mathcal{B}(\mathbf{b})$ that witnesses values $\mathbf{v} \in \mathbb{R}^k$ then there is such a profile σ where all probabilities (for any signal, action and memory mode) are at least $1/2^{2^u}$.

Proof. By Lemmas 6.3 and 6.4. □

Secondly, we show that small perturbations of probabilities only lead to small changes in (mean-payoff) values for all players.

Lemma 7.3 (small perturbations). *Consider any two profiles $\sigma, \sigma' \in \mathcal{B}(\mathbf{b})$ and let $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^k$ denote the values they witness.*

If $dist(\sigma, \sigma') < (4N)^{-1}$ then $\mathbf{v}' \in [\mathbf{v} - \gamma, \mathbf{v} + \gamma]$, where $\gamma = 9 \cdot N \cdot dist(\sigma, \sigma')C$.

Proof. After fixing the game's strategies to be σ it turns into a Markov chain M and similar let M' be the Markov chain obtained by fixing the strategies to be σ'. The state space of M and M' is the product of the states of the original game, the memory states for each player and the set of signals vectors, i.e, at most N states. Let $\overline{r}$ be the reward function in M and let $\overline{r}'$ be the one in M'.

We will apply [36, Thm. 5].This applies generally to two-player, zero-sum concurrent limit-average game s and bounds the change in (limit average) values if one changes the reward function additively and the transition function multiplicatively. Here, we apply it to the special case of Markov chains. This is because we are interested in what happens if we change the players strategies slightly (which corresponds to changing the transition and cost functions slightly as we will see). The following claim states that the reward functions and transition functions differ additively and multiplicatively respectively in the two chains.

Claim 7.4. *Let $(v, \mathbf{m}, s) \to (v', \mathbf{m}', s')$ be an edge of the Markov chains M and M'.*

- *The difference in rewards in the edge $(v, \mathbf{m}, s) \to (v', \mathbf{m}', s')$ in M and M' is at most $C \cdot \delta(\sigma, \sigma')$.*
- *Let P and P' be the probability to go from $(v, \mathbf{m}, s)$ to $(v', \mathbf{m}', s')$ in M and M' respectively. Then, $\delta(P, P') \leq \delta(\sigma, \sigma')$.*

The proof of Claim 7.4 can be found in Appendix C.1.

We can now apply [36, Thm 5], using the claim above and that the maximum absolute reward in M and M' are bounded by C, and the number of states in M and M' are N. We get that the value of M differs from the value of M' by at most $Y \stackrel{\text{def}}{=} \frac{4 \cdot N \cdot \delta(\sigma,\sigma')}{1-2 \cdot N \cdot \delta(\sigma,\sigma')} C + C \cdot \delta(\sigma, \sigma')$. Since we assume that $\delta(\sigma, \sigma') < (4N)^{-1}$, we have that $Y < 9 \cdot N \cdot \delta(\sigma, \sigma')C$. □

Lemma 7.5 (small candidates). *There exists $u \in \mathbb{N}$, polynomial in the size of the game and ε, so that the following is true.*

1. *If there exists a ε-NE σ wrt $\mathcal{B}(\mathbf{b})$ then there exists a 2ε-NE σ' wrt $\mathcal{B}(\mathbf{b})$ that is u-bit representable .*
2. *If there exists a strategy σ for player j that ε-achieves v wrt $\mathcal{B}(\mathbf{b})$ then there exists a strategy σ' for player j that 2ε-achieves v wrt $\mathcal{B}(\mathbf{b})$ and is u-bit representable.*

Proof. By Lemmas 7.2 and 7.3. □

Lemma 7.6 (Best response). *Given a game G, an $\varepsilon > 0$, an u-bit represented strategy profile and a player i, there is a u' polynomial in the size of the game, ε and u, and a u'-bit represented strategy σ' for player i that ε-achieves v wrt $\mathcal{B}(\mathbf{b})$ if there is a strategy for player i that witnesses value v wrt $\mathcal{B}(\mathbf{b})$*

Proof. For a strategy σ_i for player i, let

$$\sigma[\sigma_i] = (\sigma[1], \sigma[2], \ldots, \sigma[i-1], \sigma_i, \sigma[i+1], \ldots, \sigma[k]).$$

Given a u-bit represented strategy profile σ we can write a FO($\mathbb{R}$) expression for the strategy σ^* that maximizes the outcome for player i of playing $\sigma[\sigma^*]$ in G.

Specifically, that expression could be

$$\exists \sigma^*, y^* \forall \sigma'', y'' : \mathrm{Strat}_i(\sigma^*) \wedge \mathrm{Val}_i(\sigma[\sigma^*], y^*) \wedge ((\mathrm{Strat}_i(\sigma'') \wedge \mathrm{Val}_i(\sigma[\sigma'], y'')) \Rightarrow y^* \geq y''),$$

assuming we can express the strategy profile σ in a polynomial-sized FO($\mathbb{R}$) expression. We do that as follows: For each probability p used by σ, it is expressed in u-bit representation, $x \cdot 2^i$, where x and i are a u-bit numbers (and i is not positive, because it is a probability). We can simply write x down, but we need to be more careful with 2^i. We can write down the expressions

$$v_1 = 1/2 \wedge v_2 = v_1 \cdot v_1 \wedge v_3 = v_2 \cdot v_2 \wedge \cdots \wedge v_u = v_{u-1} \cdot v_{u-1}$$

Observe that $v_\ell = 2^{-2^\ell}$. Let i_j be the j-th bit set to 1 in the binary expression of i (clearly $j \leq u$) and there are K such bits set for some K. We then see that

$$2^i = v_{i_1} \cdot v_{i_2} \cdot \cdots \cdot v_{i_K}.$$

We can therefore write p as $x \cdot v_{i_1} \cdot v_{i_2} \cdot \cdots \cdot v_{i_K}$ (using another K variables, we can also make these products binary) and we get the full FO($\mathbb{R}$) expression by simply doing this for each of the probabilities.

Similar to Lemma 7.2, this lets us lower bound the probabilities used for strategy σ^*, by $1/2^{2^{u'}}$ for some polynomial u'. We can also apply Lemma 7.3, similar to Lemma 7.5, from which it follows that there is a u'-bit representation of a strategy σ' that ε-achieves v. □

7.3 Computing Approximations for Markov chains

The second ingredient in proving Theorem 3.3 is to approximate mean-payoff values for Markov chains in polynomial time.

Recall that fixing a strategy profile $\sigma \in \mathcal{B}(\mathbf{b})$ for a game yields a finite Markov chain where the transition probabilities encode the

players' choices and edges dictate the stepwise rewards for all players. We aim to approximate the expected mean-payoff $\mathbb{E}^{\sigma}_{v_0,s_0}\left[\underline{MP}[i]\right]$ for a given player $i \le k$, initial state v_0 and signal s_0.

Towards this, consider a Markov chain as discussed in Section 5, with states $\{1,\dots,n\}$, and where $p(i,j), r(i,j) \in \mathbb{R}$ and $d(i,j) \in \mathbb{R}$ denote the probability, reward (for the chosen player), and duration of an edge $i \to j$. Recall that it suffices to compute the expected mean-payoff ratio (see Equation (1)) as initially, the duration of every step is 1.

We propose an algorithm that exhaustively applies the loop and state-elimination procedures of Definitions 5.1 and 5.3, which is correct by Lemmas 5.2 and 5.4. However, to deal with the necessarily double-exponentially small values in input and output Markov chains, we replace the precise arithmetic operands by approximate ones that use floating point representations with polynomial many bits. We show that the error this introduces can be bounded polynomially.

Definition 7.7. A Markov chain is *represented in u-bits* if all probabilities, rewards and durations are in $\mathbb{Q}(u)$ and given in u-bit floating point representation.

In particular, all distributions $(p_{ij})_{j\in V}$ of a state i is in $\mathbb{P}(u)$ given in u-bit approximately normalised representation using u-bit floating point numbers.

We now state the bounds on approximate variants of state and loop eliminations. Let $\oplus^u, \oslash^u, \otimes^u$ be the u-bit finite precision variant of addition, division and multiplication, respectively. These map values $\mathbb{Q}(u)^2$ to $\mathbb{Q}(u)$ by rounding down the result of the corresponding arithmetic operation to the nearest value in $\mathbb{Q}(u)$. We drop the superscript u for readability.

Lemma 7.8 (State Elimination). *There is a polynomial δ_1 so that the following holds for all $u \in \mathbb{N}$.*

Let $M = (V,p,r,d)$ and $M' = (V,p',r',d')$ be Markov chains represented in u-bits so that M' results from M by eliminating a transient state n as in Definition 5.3 but using u-bit floating point arithmetic. That is, for all $i,j \neq n$,

$$\begin{aligned} p'(i,j) &= p(i,j) \oplus (p(j,n) \otimes p(n,j)) \\ r'(i,j) &= (p(i,j) \otimes r(i,j)) \\ &\quad \oplus ((p(i,n) \otimes p(n,j)) \otimes (r(i,n) \oplus r(n,j))) \\ d'(i,j) &= (p(i,j) \otimes d(i,j)) \\ &\quad \oplus ((p(i,n) \otimes p(n,j)) \otimes (d(i,n) \oplus d(n,j))) \end{aligned}$$

Then,

1. *The smallest (negative) exponent among all floating point numbers in M' is at most one smaller than that in M.*
2. *Then $\left|\mathbb{E}^{M}_{1}[MPR] - \mathbb{E}^{M'}_{1}[MPR]\right| \le \delta_1 2^{-u}$.*

Lemma 7.9 (Loop Elimination). *There is a polynomial δ_2 so that the following holds for all $u \in \mathbb{N}$.*

Let $M = (V,p,r,d)$ and $M' = (V,p',r',d')$ be Markov chains represented in u-bits so that M' results from M by eliminating a loop in state n as in Definition 5.3 but using u-bit floating point arithmetic. That is, $p'(n,n) = 0$; and for all $i,j \neq n$ let $p'(i,j) = p(i,j)$, $r'(i,j) = r(i,j)$, $d'(i,j) = d(i,j)$ and

$$\begin{aligned} p'(n,j) &= p(n,j) \oslash (\oplus_{k\neq i} p(n,k)) \\ r'(n,j) &= ((r(n,j) \otimes p(n,j)) \oplus (p_{ii} \otimes r_{ii})) \\ &\quad \oslash (\oplus_{k\neq i} p(n,k)) \\ d'(n,j) &= ((d(n,j) \otimes p(n,j)) \oplus (p(n,n) \otimes d(n,n))) \\ &\quad \oslash (\oplus_{k\neq i} p(n,k)). \end{aligned}$$

Then,

1. *The smallest (negative) exponent among all floating point numbers in M' is at most one smaller than that in M.*
2. *Then $\left|\mathbb{E}^{M}_{1}[MPR] - \mathbb{E}^{M'}_{1}[MPR]\right| \le \delta_2 2^{-u}$.*

We now show that one can approximate the mean-payoff ratio in polynomial time and sufficiently closely.

Lemma 7.10. *There is a polynomial time algorithm for the following problem. Given a Markov chain with n states represented in u-bits and error $\varepsilon \in (0,1)$ in binary.*

Output a value in $[v, v+\varepsilon)$, where $v = \mathbb{E}^{M}_{1}[MPR]$ is the expected mean-payoff value from the initial state of M.

Proof. W.l.o.g., we can assume that $u \ge 1000n^2$ and $\varepsilon > n(\delta_1+\delta_2)2^{-u}$. Otherwise, one can increase u to $\log_2((\delta_1+\delta_2)n)\cdot\ell$, where ℓ is the number of bits used in the denominator of ε, i.e, $\varepsilon \ge 1/2^{\ell}$. Since all the constants used are polynomial in the size of the input, so will be u.

The algorithm will alternate between

1. exhaustively applying Lemma 7.9 to remove self-loops in all transient states, and
2. applying Lemma 7.8 to remove a transient state $n \neq 1$.

This will result in a Markov chain M' where each original CCS is collapsed into a single absorbing state. Now, if M' contains only a single absorbing state s, then $\mathbb{E}^{M'}_{1}[MPR] = r'(s,s)/d'(s,s)$, so we output $r'(s,s) \oslash d'(s,s)$. Otherwise, 1 is the only transient state in M' and we output $\oplus_{i\neq 1} p'(1,i)\ (r'(i,i) \oslash d'(i,i))$.

Notice that at most $n-1$ many transient states can be removed, which bounds the number of times we invoke Lemma 7.8. Similarly, in between removing states, step (1) results in at most n many applications of Lemma 7.9.

Therefore, by Lemma 7.9(1) and Lemma 7.8(1), the largest exponent in any floating point representation of M' has at most increased by $n(n+1)$ compared to M. Moreover, the total error introduced by these operations is at most

$$\left|\mathbb{E}^{M}_{1}[MPR] - \mathbb{E}^{M'}_{1}[MPR]\right| \le n(\delta_1+\delta_2)2^{-u})$$

by Lemma 7.9(2) and Lemma 7.8(2).

By plugging in the value of u, we get the error is at most $2^{-\ell}$, therefore at most ε. □

7.4 Proof of Theorem 3.3

We are now ready to prove Theorem 3.3.

Theorem 3.3. *For every fixed $k \ge 1$ the following are in $\mathsf{FNP}^{\mathsf{NP}}$.*

1. *Given a k-player game, bounds $b \in \mathbb{N}^k$ and $\varepsilon \in (0,1/2)$, find an ε-Nash equilibrium with respect to $\mathcal{B}(\mathbf{b})$ (if for some $\varepsilon' \in [0,\varepsilon/3)$, an ε'-Nash equilibrium with respect to $\mathcal{B}(\mathbf{b})$ exists);*

(2) *Given a 2-player zero-sum game, bounds $b \in \mathbb{N}^k$, $\varepsilon \in (0, 1/2)$ and $v \in \mathbb{R}$, find a strategy that ε-achieves v with respect to $\mathcal{B}(\mathbf{b})$ (if for some $\varepsilon' \in [0, \varepsilon/3)$, some strategy ε'-achieves v with respect to $\mathcal{B}(\mathbf{b})$).*

We therefore want to make two algorithms in $\mathsf{FNP}^{\mathsf{NP}}$. Because the algorithms are quite similar, we state them both first before proving correctness.

First algorithm (for (1)). First, for (1), we want to find a 3ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$ (if an ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$ exists). To do so, we use an oracle that solves the following question: Given a game G, $\varepsilon > 0$ and an u-bit represented strategy profile σ, guess an u'-bit represented strategy profile σ'. For each player i, construct the induced Markov chain MC_1^i for σ and the induced Markov chain MC_2^i for $(\sigma[1], \ldots, \sigma[i-1], \sigma'[i], \sigma[i+1], \ldots, \sigma[k])$, each with rewards for player i. Apply the algorithm from Lemma 7.10, with the value of ε being $\varepsilon/3$ and u being u', to each Markov chain, giving approximated values v_1^i and v_2^i respectively. If $v_2^i - v_1^i > \frac{7}{3}\varepsilon$, then return "yes". If "yes" has not been returned for any player, return "no".

The $\mathsf{FNP}^{\mathsf{NP}}$algorithm (for (1)) that uses the oracle is then as follows: Given a game G and an $\varepsilon > 0$, guess an u-bit represented strategy profile σ and apply the oracle with G, ε and σ. If the oracle returns "yes", return "no" and otherwise return σ.

Second algorithm (for (2)). Next, for (2), given a value v, we want to find a strategy that 3ε-achieves v wrt $\mathcal{B}(\mathbf{b})$ (if a strategy that ε-achieves v wrt $\mathcal{B}(\mathbf{b})$ exists). To do so, we use an oracle that solves the following question: Given a game G, $\varepsilon > 0$ and an u-bit represented strategy σ for player 1, guess an u'-bit represented strategy σ' for player 2. Apply the algorithm from Lemma 7.10, with ε being $\varepsilon/3$ and u being u', to the Markov chain induced by (σ, σ'), given an approximated value v'. If $v - v' > \frac{7}{3}\varepsilon$, then return "yes", otherwise, return "no".

The $\mathsf{FNP}^{\mathsf{NP}}$algorithm (for (2)) that uses the oracle is then as follows: Given a game G and an $\varepsilon > 0$, guess an u-bit represented strategy σ and apply the oracle with G, ε and σ. If the oracle returns "yes", return "no" and otherwise return σ.

Running time and correctness of the algorithms. Because the running time of the algorithm from Lemma 7.10 is polynomial and every u-bit represented strategy is also of polynomial size, the oracles are in NP and the algorithms are in $\mathsf{FNP}^{\mathsf{NP}}$.

We thus just need to argue that the algorithms are correct.

Lemma 7.11. *We have the following:*

(1) *The first algorithm (for (1)) returns either "no" or a 3ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$. Also, if an ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$ exists it will not return "no".*
(2) *The second algorithm (for (2)) returns either "no" or a strategy that 3ε-achieve v wrt $\mathcal{B}(\mathbf{b})$. Also, if a strategy that ε-achieve v wrt $\mathcal{B}(\mathbf{b})$ exists it will not return "no".*

Proof. First for (1): We start by arguing that if the algorithm guessed an u-bit representable 2ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$, σ, which exists if an ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$ exists by Lemma 7.5, then the oracle will necessarily return "no" and therefore such a strategy profile can be returned. Consider first that if we use the real value of MC_1^i, then it must be at least the real value of $MC_2^i - 2\varepsilon$, because σ is a 2ε-Nash equilibrium. The approximated value of MC_1^i is at most $\varepsilon/3$ smaller than the real value. Similar for MC_2^i. This means that the approximated value of MC_1^i must be at least the approximated value of $MC_2^i - \frac{7}{3}\varepsilon$ and therefore the oracle will necessarily return "no". Because at least one guess would result in an output different from "no", the algorithm must return some such value.

Next, consider that the algorithm guessed some u-bit representable strategy profile σ, which was not a 3ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$ and we will argue that the oracle will be able to return "yes". Let v be the outcome of σ for player i. By definition of 3ε-Nash equilibrium wrt $\mathcal{B}(\mathbf{b})$, there must be some player i and strategy wrt $\mathcal{B}(\mathbf{b})$, σ'', for player i, so that value v'' of the Markov chain induced by $(\sigma[1], \ldots, \sigma[i-1], \sigma'', \sigma[i+1], \ldots, \sigma[k])$ with rewards for player i must be strictly greater than $v + 3\varepsilon$. By Lemma 7.6, applied with ε being $\varepsilon/3$ we can then find an u'-bit representable strategy σ''' so that $(\sigma[1], \ldots, \sigma[i-1], \sigma''', \sigma[i+1], \ldots, \sigma[k])$ with rewards for player i must have a value strictly greater than $v + \frac{8}{3}\varepsilon$. The oracle can then non-deterministically guess a strategy profile in which $\sigma'[i] = \sigma'''$, the approximated values v_1^i and v_2^i would then be such that $v_2^i - v_1^i > \frac{7}{3}\varepsilon$ (because each is at most $\varepsilon/3$ smaller than the real value) and thus, the oracle will return "yes".

Next, we will argue that the algorithm for (2) is correct. The argument is similar to the above argument but included for completeness. We start by arguing that if the algorithm guessed a u-bit represented strategy σ that 2ε-achieves v wrt $\mathcal{B}(\mathbf{b})$ (which exists if a strategy that ε-achieves v wrt $\mathcal{B}(\mathbf{b})$ exists by Lemma 7.5), then the oracle will return "no" and therefore, the algorithm can return such σ. Consider first that if we use the real value of the induced Markov chain, then it must be at least $v - 2\varepsilon$, because the strategy σ 2ε-achieves v. Because we used an approximation of that value, called v', that is at most $\varepsilon/3$ smaller than the real value, we get that $v \leq v' + \frac{7}{3}\varepsilon$ and thus the oracle will necessarily return "no". Because at least one guess would result in an output different from "no", the algorithm must return some such value.

Next, consider that the algorithm guessed some u-bit representable strategy σ, which did not 3ε-achieve v wrt $\mathcal{B}(\mathbf{b})$ and we will argue that the oracle will be able to return "yes". By definition of 3ε-achieve v wrt $\mathcal{B}(\mathbf{b})$, there must be some strategy σ'' wrt $\mathcal{B}(\mathbf{b})$ for player 2, so that value v'' of the Markov chain induced by (σ, σ'') with rewards for player 1 must be strictly smaller than $v - 3\varepsilon$. By Lemma 7.6, applied with ε being $\varepsilon/3$ we can then find an u'-bit representable strategy σ''' so that (σ, σ''') with rewards for player 1 must have a value strictly smaller than $v - \frac{8}{3}\varepsilon$. The oracle can non-deterministically guess that strategy in which $\sigma' = \sigma'''$, the approximated value v' is then such that $v > v' + \frac{7}{3}\varepsilon$ (because the approximated value is at most $\varepsilon/3$ smaller than the real value) and thus, the oracle will return "yes". □

8 APPLICATIONS

We discuss the implications of our results for some well-known classes of games.

8.1 Multi-player partial-information parity games

A *parity* objective (for player $j \leq k$) assigns a play $\rho = (v_0, s_0, a_0, c_0), (v_1, s_1, a_1, c_1) \dots$ the value 1 if $\limsup_{N\to\infty} c_i[j]$, the maximal reward seen infinitely often, is even and 0 otherwise.

In perfect-information turn-based games, this condition can be translated into a mean-payoff condition, based on the observation that the game effectively ends after a cycle is formed (see e.g. [18, Theorem 40]). However, no such simple reduction can generalise even to concurrent games, because the most significant reward may occur arbitrarily rarely. In concurrent games, as opposed to the simpler turn-based ones, ε-optimal strategies might require infinite memory [25] and mixed actions with double exponentially small probabilities [27].

We now argue that one can adapt our constructions to parity objectives as follows. First, state and loop elimination (Section 5) are simpler for parity since one does not need to keep track of the duration or of the exact reward but can instead just set the new reward to be the maximum of the old ones.

The only non-trivial change required is the proof of Lemma 7.3, which bounds the change in values achieved by strategy profiles under small perturbations. Specifically, our argument relies on [36, Theorem 5], which only applies to mean-payoff games and not parity games. To adapt our proof, we consider the Markov chain induced by any fixed strategy profile where each player uses finite memory. Now almost surely, some CCS is entered and then every reward contained is seen infinitely often. Therefore a player gets overall parity reward 1 in a CCS within which their largest reward is even (and 0 otherwise). Therefore, for each CCS, any perturbation of the probabilities in the corresponding states (i.e. state in the original game and memory vector) that have the same support will not change the outcome of that CCS. We can therefore define an equivalent mean-payoff Markov chain in which each CSS is collapsed to an absorbing chain with reward 1 or 0 accordingly. The claim of Lemma 7.3 (parity) then follows from Lemma 7.3 as stated for mean-payoff.

We conclude that Theorems 3.2 and 3.3 hold as stated also for (multi-player, partial-information) games with parity objectives.

8.2 Multi-player concurrent games

We consider different classes of perfect-information games that are *concurrent* games, meaning the motion is determined by simultaneously chosen actions of all players, as opposed to the *turn-based* games where in each round only the choice of one player matters.

When considering bounded strategies, it is instructive to distinguish *private* memory (accessible only by the respective player as used throughout this paper) from *public* memory, which is visible to all players. Essentially, public memory is shared among all players (but can only be updated by its owner) (see [25, Section 5.1] for details).

The important difference for us is that fixing public memory strategies for all but one player results in an ordinary MDP (a perfect-information 1-player game) for the remaining player. On the other hand, fixing private memory strategies for all but one player, results in a *partial-information* MDP, as the player cannot know their opponents' memory modes.

8.2.1 Computing ε-Nash Equilibria. Our technique allows us to compute general ε-Nash equilibria, i.e., without restricting the set of admissible strategies, assuming that there exist equilibria with bounded public memory. Formally, for a k-player game and vector $\mathbf{b} \in \mathbb{N}^k$, let $\mathcal{P}(\mathbf{b})$ denote the set of all strategy profiles wherein each player i uses a public finite memory strategy with at most $\mathbf{b}[i]$ modes.

Corollary 8.1. *For any fixed $k \geq 1$, we get the following: Given a k-player concurrent mean-payoff game, bounds $\mathbf{b} \in N^k$ (in unary), and $\varepsilon > 0$ (in binary), we can*

(1) find an ε-Nash equilibrium (and output it) in exponential time, if an ε-Nash equilibrium wrt $\mathcal{P}(\mathbf{b})$ exists
(2) find an ε-Nash equilibrium (and output it) in $\mathsf{FNP}^{\mathsf{NP}}$, if there exists an ε'-Nash equilibrium wrt $\mathcal{P}(\mathbf{b})$, for some $\varepsilon' \in [0, \varepsilon/2)$.

Proof. This follows from Theorems 3.2 and 3.3, using the observation that any ε-Nash equilibrium in $\mathcal{P}(\mathbf{b})$ is also an ε-Nash equilibrium without restriction, ie., where unilateral deviations to any strategy are considered, because of the following. Each player i, no matter which bounded and public memory strategy each of the other players picked, player i cannot deviate to a bounded and public memory strategy ensuring more than ε than what player i received before, by definition of ε-Nash equilibria with respect to $\mathcal{P}(\mathbf{b})$. Finite MDPs with mean-payoff objectives have memoryless ε-optimal strategies [33]. Therefore, player i cannot get more than what can be ensured by a memoryless strategy which is a special case of bounded and public memory strategies. □

It is an open question whether ε-Nash equilibria always exist for concurrent mean-payoff games. We point to two special cases: stay-in-a-set games [34] and quitting games [37, 38]. For both, it was known that finite (and as we will argue: public) memory ε-Nash equilibria exist, but so far, no algorithm was known to compute them.

8.2.2 Stay-In-a-set games. These are multi-player concurrent games where each player has a separate safety objective, that is, wants the play to remain in a given subset of the states. Secchi and Sudderth [34] showed that stay-in-a-set games always have finite memory ε-Nash equilibria, but the players might need to remember who has already lost. Such strategies are finite memory public strategies (with the number of memories equal to 2^k, where k is the number of players). For a fixed number of players k, our technique can be used to compute ε-Nash equilibria with strategies using public memory and optimise the necessary memory bounds. That is, we can find public-memory equilibria with the least amount of memory.

Corollary 8.2. *For every $k \geq 1$ the following is in $\mathsf{FNP}^{\mathsf{NP}}$. Given a k-player stay-in-a-set game, find an ε-Nash equilibria.*

Proof. One first encodes the safety conditions as a mean-payoff condition, so that a player receives reward 1 as long as their set is never left, and 0 forever once it is. Using a subset construction, this results in a game with states $V \times 2^k$ (a polynomial blow-up since k is fixed). In order to keep a one-to-one correspondence of equilibria it is necessary to construct a partial-information game here. In particular, in the original game, it is sufficient for players

to remember who has already lost. Our reduction encodes this information into the states, and so it must be hidden from players (by having any state $(s, \mathbf{m})$ give the signal s) because otherwise, memoryless strategies suffice, even if they did not in the original game. The claim now follows by Theorem 3.3, bounding the number of memory modes for each player by $b = 2^k$, similar to the proof of Corollary 8.1, because, if each player is playing a public and finite-memory strategy, one just needs $\mathbf{m}$ in memory (requiring 2^k memory states) to have all information about the state of the game. □

8.2.3 Quitting games. Quitting games [37, 38] are concurrent mean-payoff games in which there is only one non-absorbing state and the players each have two actions in that state: Continue or Stop. The play goes to an absorbing state (effectively stops) in the first round when at least one player plays their stop action.

Under mild assumptions on the payoffs[2], Solan and Vieille [37] show the existence of "cyclic" ε-NE, in which players' behaviour repeat after some number of stages. Any such equilibrium can be represented as one consisting of public finite memory strategies. Specifically, we need to have a state of memory corresponding to each stage in the "cyclic" ε-NE, from stage one up to the first repetition, where we just in each stage move to the next memory state and then loop back at the end. While they do not provide a bound directly, they prove the existence of such 'cyclic" ε-NE and their proof implies a bound: The proof of [37, Prop. 2.3] uses a partitioning of $[-R, R]^k$ (if each reward is bounded absolutely by R) into regions, so that any two elements in one region differ by no more than ε^2 (in one-norm). For fixed k, this gives on the order of $O\left(\frac{R^k}{\varepsilon^{2k}}\right)$ many regions. Each region is then mapped to some (perhaps other) region and the cyclic ε-Nash equilibrium is then formed from the sequence obtained by the regions encountered by repeatedly following the mapping. The length of such a sequence can therefore not exceed the number of encountered regions, bounded as above.

Our algorithm can therefore find such an ε-Nash equilibrium, using Corollary 8.1, assuming that both rewards and ε are encoded in unary.

Corollary 8.3. *For any fixed $k \geq 1$, we get the following: Given a k-player quitting game (with rewards in unary), satisfying the conditions in [37], and $\varepsilon > 0$ (in unary), we can find an ε-Nash equilibrium (and output it) in* $\mathsf{FNP}^{\mathsf{NP}}$.

Similarly, given a game and a cyclic ε-Nash equilibrium, one can compute a finite memory ε-Nash equilibrium that uses less memory if it exists.

8.3 2-player, zero-sum concurrent mean-payoff games

8.3.1 Checking the existence of memoryless (ε-)optimal strategies. Recent work by Bordais et al. [5–7] studies how to restrict the topology of the game graph to ensure the existence of memoryless optimal strategies for both players. They show that it is decidable (in exponential time) whether their properties hold, and thus ε-optimal strategies exist. The constructions are based on effective reductions to the first-order theory of the reals, which also yields an exponential time algorithm to compute (ε-)optimal memoryless strategies if they exist. This has been done for reachability [6] Büchi and coBüchi [5], and parity [7] conditions.

We improve on these results in several ways: Our constructions are directly applicable to not only zero-sum concurrent parity games (see Section 8.1) but to more general settings, such as with mean-payoff objectives or where bounded-, finite- and public-memory strategies are sought instead of just memoryless ones. We can decide the existence of (ε-)optimal memoryless strategies for both players without the proxy of deciding sufficient criteria. Finally, for $\varepsilon > 0$, our technique results in a lower complexity: $\mathsf{FNP}^{\mathsf{NP}}$ instead of exponential time.

Corollary 8.4. *For two-player, zero-sum concurrent mean-payoff games,*

1. *checking if, for any given game and $\varepsilon \geq 0$, there exist ε-optimal memoryless strategies for both players (and output one for each player if so) can be done in exponential time.*
2. *checking if, for any given game and $\varepsilon > 0$, there exist ε-optimal memoryless strategies for both players (and output one for each player if so), if there exists an ε'-optimal memoryless strategy for each player, for some $\varepsilon' \in [0, \varepsilon/2)$ is in* $\mathsf{FNP}^{\mathsf{NP}}$.

Proof. If you have an $\varepsilon/2$-optimal strategy for each player in a zero-sum, two-player game, then the strategies form an ε-Nash equilibrium and given an ε-Nash equilibrium, it must be made up of two strategies that are each ε-optimal. Because the strategies are memoryless ($b = 1$), they are in particular public memory. The two claims thus follow by Corollary 8.1 instantiated for $k = 2$ and $b = 1$. □

Approximating zero-sum values. We consider the problem of approximating ordinary (zero-sum) values for two-player concurrent mean-payoff games. Recall that all strategies getting close to ensuring the value might require infinite memory [25].

Different methods (see eg. [13, 26, 31, 32]) have been proposed for this. The most well-known approach is the value iteration algorithm of Mertens and Neyman [31], which has later been show to take double-exponentially many iterations to ε-approximate the values, see [26, 27]. The current best complexity upper bound is PSPACE: As shown by [26, 31] using a double exponentially small discount factor, the corresponding discounted game to a mean-payoff game will have the same value $\pm\varepsilon$. As shown by [16] a discounted game can be solved in PSPACE by reduction[3] to the existential fragment of $\mathrm{FO}(\mathbb{R})$.

Our technique yields the current best algorithm for this problem.

Corollary 8.5. *The following is in $\mathsf{FNP}^{\mathsf{NP}}$. Given a two-player, zero-sum concurrent mean-payoff game and $\varepsilon > 0$, find the value of the game within an additive error of ε.*

Proof. As shown already in [31], the value of a mean-payoff game is equal to the limit of values of the same game but with discount d for d going to 0. Hansen et al. [26] show that for any

[2] They ask that (i) Each player prefers to be the only quitter rather than having the game continue forever, and (ii) if a player quits, he prefers to be the only quitter.

[3] Etessami and Yannakakis [16] show how to construct $\exists\mathrm{FO}(\mathbb{R})$ formulae for games with fixed discount factors, but one can create double-exponential values using repeated squaring in the existential fragment.

discount d below some double exponentially small number, the value of the mean-payoff game differs by less than ε from that of the discounted game with discount factor d. For a formal definition of discounted-payoff games, we refer to [26, 37].

There is a well-known reduction from discounted games to (non-discounted) mean-payoff games already present in [35]: Given a d-discounted game G, replace each action a from s to t with a stochastic action that goes to an absorbing state with reward $r(a)$ with probability d and otherwise (with the remaining probability of $1-d$) proceeds to t (it works the same way if a was part of a stochastic action: If it occurred with probability p, then you instead go to the absorbing state with probability pd and to t with probability $p(1-d)$). This results in a mean-payoff game G', with transition probabilities in the order of d. Importantly, the value and witnessing strategies are the same in both games. As any concurrent discounted game has optimal memoryless strategies for both players[35], the same is true for G'. Such strategies are in particular public memory strategies. As is well-known, in a two-player, zero-sum game, a Nash equilibrium must be made up of two optimal strategies and an optimal strategy for each player forms an Nash equilibrium.

The claim now follows from Corollary 8.1. Note that the corollary is applicable even though d is double-exponentially small because our algorithm allows transition probabilities to be represented in floating point notation. □

REFERENCES

[1] Yackolley Amoussou-Guenou, Souheib Baarir, Maria Potop-Butucaru, Nathalie Sznajder, Léo Tible, and Sébastien Tixeuil. 2021. On the Encoding and Solving of Partial Information Games. In *Networked Systems*. 60–76. https://doi.org/10.1007/978-3-030-67087-0_5

[2] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. 2006. *Algorithms in Real Algebraic Geometry* (2nd ed.). https://doi.org/10.1007/3-540-33099-2

[3] Patrick Billingsley. 1995. *Probability and Measure*.

[4] David Blackwell and Tom S Ferguson. 1968. The Big Match. *The Annals of Mathematical Statistics* 39, 1 (1968), 159–163. https://doi.org/10.1214/aoms/1177698513

[5] Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. 2022. Playing (Almost-)Optimally in Concurrent Büchi and Co-Büchi Games. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, Vol. 250. 33:1–33:18. https://doi.org/10.4230/LIPIcs.FSTTCS.2022.33

[6] Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. 2021. From Local to Global Determinacy in Concurrent Graph Games. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS) (LIPIcs, Vol. 213)*. 41:1–41:14. https://doi.org/10.4230/LIPIcs.FSTTCS.2021.41

[7] Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. 2023. From Local To Global Optimality in Concurrent Parity Games. arXiv:2311.14373 [cs.GT]

[8] Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. 2021. Arena-Independent Finite-Memory Determinacy in Stochastic Games. In *International Conference on Concurrency Theory (CONCUR) (LIPIcs, Vol. 203)*. 26:1–26:18. https://doi.org/10.4230/LIPIcs.CONCUR.2021.26

[9] Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. 2023. Characterizing Omega-Regularity through Finite-Memory Determinacy of Games on Infinite Graphs. *TheoretiCS* 2 (2023). https://doi.org/10.46298/theoretics.23.1

[10] Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. 2022. Games Where You Can Play Optimally with Arena-Independent Finite Memory. *Logical Methods in Computer Science* 18, 1 (2022). https://doi.org/10.46298/lmcs-18(1:11)2022

[11] A.L. Cauchy. 1828. *Exercices de Mathematiques (1828)*.

[12] Krishnendu Chatterjee, Laurent Doyen, Sumit Nain, and Moshe Y. Vardi. 2014. The Complexity of Partial-Observation Stochastic Parity Games with Finite-Memory Strategies. In *International Conference on Foundations of Software Science and Computational Structures (FoSSaCS)*. 242–257. https://doi.org/10.1007/978-3-642-54830-7_16

[13] Krishnendu Chatterjee, Rupak Majumdar, and Thomas Henzinger. 2008. Stochastic limit-average games are in EXPTIME. *International Journal of Game Theory* 37 (06 2008), 219–234. https://doi.org/10.1007/s00182-007-0110-5

[14] Krishnendu Chatterjee, Raimundo Saona, and Bruno Ziliotto. 2022. Finite-Memory Strategies in POMDPs with Long-Run Average Objectives. *Mathematics of Operations Research* 47, 1 (feb 2022), 100–119. https://doi.org/10.1287/moor.2020.1116

[15] Andrzej Ehrenfeucht and Jan Mycielski. 1973. Positional games over a graph. *Notices of the American Mathematical Society* 20 (1973), A–334. https://doi.org/10.1007/BF01768705

[16] Kousha Etessami and Mihalis Yannakakis. 2008. Recursive Concurrent Stochastic Games. *Logical Methods in Computer Science* 4, 4 (11 Nov. 2008). https://doi.org/10.2168/LMCS-4(4:7)2008

[17] Hugh Everett. 1957. Recursive games. In *Contributions to the Theory of Games, Volume III*. Annals of Mathematics Studies, Vol. 39. 47–78. https://doi.org/10.1515/9781400829156-014

[18] Nathanaël Fijalkow and Benjamin Monmege. 2023. Games with Payoffs. In *Games on Graphs*. https://doi.org/10.48550/arXiv.2305.10546

[19] Jerzy A. Filar, Todd A. Schultz, Frank Thuijsman, and O. J. Vrieze. 1991. Nonlinear programming and stationary equilibria in stochastic games. *Math. Program.* 50 (1991), 227–237. https://doi.org/10.1007/BF01594936

[20] Søren Kristoffer Stiil Frederiksen and Peter Bro Miltersen. 2013. Approximating the Value of a Concurrent Reachability Game in the Polynomial Time Hierarchy. In *International Symposium on Algorithms and Computation (ISAAC)*. 457–467. https://doi.org/10.1007/978-3-642-45030-3_43

[21] Dean Gillette. 1957. Stochastic games with zero stop probabilities. *Contributions to the Theory of Games* 3 (1957), 179–187. https://doi.org/10.1515/9781400882151-011

[22] Hugo Gimbert and Wiesław Zielonka. 2005. Games Where You Can Play Optimally Without Any Memory. In *International Conference on Concurrency Theory (CONCUR)*. 428–442.

[23] Oded Goldreich. 2006. *On Promise Problems: A Survey*. 254–290. https://doi.org/10.1007/11685654_12

[24] Kristoffer Arnsfelt Hansen, Rasmus Ibsen-Jensen, and Peter Bro Miltersen. 2014. The Complexity of Solving Reachability Games Using Value and Strategy Iteration. *Theory Comput. Syst.* 55, 2 (2014), 380–403. https://doi.org/10.1007/s00224-013-9524-6

[25] Kristoffer Arnsfelt Hansen, Rasmus Ibsen-Jensen, and Abraham Neyman. 2023. The Big Match with a Clock and a Bit of Memory. *Mathematics of Operations Research* 48, 1 (2023), 419–432. https://doi.org/10.1287/moor.2022.1267

[26] Kristoffer Arnsfelt Hansen, Michal Koucky, Niels Lauritzen, Peter Bro Miltersen, and Elias P. Tsigaridas. 2011. Exact Algorithms for Solving Stochastic Games: Extended Abstract. In *Symposium on Theory of Computing (STOC)*. 205–214. https://doi.org/10.1145/1993636.1993665

[27] Kristoffer Arnsfelt Hansen, Michal Koucky, and Peter Bro Miltersen. 2009. Winning Concurrent Reachability Games Requires Doubly-Exponential Patience. In *ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 332–341. https://doi.org/10.1109/LICS.2009.44

[28] Paul Hunter, Arno Pauly, Guillermo A. Pérez, and Jean-François Raskin. 2018. Mean-payoff games with partial observation. *Theoretical Computer Science* 735 (2018), 82–110. https://doi.org/10.1016/j.tcs.2017.03.038

[29] Omid Madani, Steve Hanks, and Anne Condon. 1999. On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems. In *AAAI/IAAI*. https://doi.org/10.5555/315149.315395

[30] Yishay Mansour and Satinder Singh. 1999. On the Complexity of Policy Iteration. In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*. 401–408. https://doi.org/10.5555/2073796.2073842

[31] J-F Mertens and Abraham Neyman. 1981. Stochastic games. *International Journal of Game Theory* 10, 2 (1981), 53–66.

[32] Miquel Oliu-Barton. 2021. New Algorithms for Solving Zero-Sum Stochastic Games. *Mathematics of Operations Research* 46, 1 (2021), 255–267. https://doi.org/10.1287/moor.2020.1055

[33] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). https://doi.org/10.1002/9780470316887

[34] Piercesare Secchi and William D. Sudderth. 2002. Stay-in-a-set games. *International Journal of Game Theory* 30 (2002), 479–490. https://doi.org/10.1007/s001820200092

[35] Lloyd S. Shapley. 1953. Stochastic Games. *Proceedings of the National Academy of Sciences* 39 (1953), 1095 – 1100. https://doi.org/10.1073/pnas.39.10.1095

[36] Eilon Solan. 2003. Continuity of the Value of Competitive Markov Decision Processes. *Journal of Theoretical Probability* 16 (10 2003), 831–845. https://doi.org/10.1023/B:JOTP.0000011995.28536.ef

[37] Eilon Solan and Nicolas Vieille. 2001. Quitting Games. *Mathematics of Operations Research* 26, 2 (2001), 265–285. https://doi.org/10.1287/moor.26.2.265.10549

[38] Eilon Solan and Nicolas Vieille. 2003. Quitting Games – An Example. *International Journal of Game Theory* 31, 3 (2003), 365–381. https://doi.org/10.1007/s001820200125

[39] Eilon Solan and Bruno Ziliotto. 2016. *Stochastic Games with Signals*. 77–94.

A PROOFS FOR SECTION 5

Definition A.1 (Cycle values). For a set $C \subseteq V$ of target states let $H_C : \Omega \to \mathbb{N} \cup \{\infty\}$ be the random variable denoting the least non-zero number of steps until state in C is reached. That is, $H_C(s_0 s_1 \ldots) = \inf\{k \geq 1 \mid s_k \in C\}$. Further, let $R_t, D_t : \Omega \to \mathbb{R} \cup \{\infty\}$ denote the sum of rewards, and durations respectively, accumulated up until state t is first reached. That is, $R_t = \sum_{k=1}^{H_t} r(s_{k-1}, s_k)$ and $D_t = \sum_{k=1}^{H_t} d(s_{k-1}, s_k)$.

Lemma A.2. *If s is recurrent then $\mathbb{E}_s[MPR] = \mathbb{E}_s[R_s] / \mathbb{E}_s[D_s]$.*

Proof. Let s belong to some CCS S with n states. Let p be the smallest non-zero probability associated to any edge between vertices in S. Let R (resp. r) be the most positive (resp. most negative) reward per step and D (resp. d) be the largest (resp. smallest) duration of a step.

By definition of CCS, there is always a path between any two states in S and thus in particular, a simple one (i.e. with no state occurring twice). This means that there is a simple path consisting of at most $n-1$ edges between the states. Each edge has probability of at least p of occurring, so the path has a probability of at least p^{n-1} to be followed. From each s' consider a shortest (in number of steps) directed path $P_{s'}$ from s' to s. We start an *attempt* whenever we visit s or just after we have moved along an edge not in $P_{s'}$, where s' was the last state in which we started an attempt. In particular, we start a new attempt every at most n steps. Each attempt reaches s with probability at least p^{n-1}. Therefore, in expectation, we need at most p^{1-n} many attempts. Each attempt took at most n steps.

This means that almost all plays visits s infinitely many times. Let s_i be the random variable denoting the step in which the play visits s the ith time. We view the steps inbetween s_i and s_{i+1} as a *trial* and we thus have infinitely many trials.

Consider any $\varepsilon > 0$. We partition plays into two sets, G and N. A play is in G, if the average reward of the trials is within $(1 \pm \varepsilon)$ of $\mathbb{E}_s[R_s]$ (unless $\mathbb{E}_s[R_s] = 0$, in which case, we have that the average reward of the trials is in $[-\varepsilon, \varepsilon]$ for a play to be in G) and the average duration of the trials is within $(1 \pm \varepsilon)$ of $\mathbb{E}_s[D_s]$. The remaining plays are in N.

By linearity of expectation, we have that

$$\mathbb{E}_s[MPR] = \mathbb{E}_s[MPR \mid N]\,\mathbb{P}_s(N) + \mathbb{E}_s[MPR \mid G]\,\mathbb{P}_s(G)\,.$$

For any play, MPR is in $[\min(r/d, r/D), \max(R/d, R/D)]$, because that is the bound on the stepwise fraction. In particular, it is true for $\mathbb{E}_s[MPR \mid N]$. By law of large numbers, $\mathbb{P}_s(N) < \varepsilon$.

By definition of G we see as follows. If $\mathbb{E}_s[R_s] > 0$, we have that

$$\mathbb{E}_s[MPR \mid G] \in \left[\frac{(1-\varepsilon)\mathbb{E}_s[R_s]}{(1+\varepsilon)\mathbb{E}_s[D_s]}, \frac{(1+\varepsilon)\mathbb{E}_s[R_s]}{(1-\varepsilon)\mathbb{E}_s[D_s]}\right]$$

Conversely, if $\mathbb{E}_s[R_s] < 0$, we have that

$$\mathbb{E}_s[MPR \mid G] \in \left[\frac{(1+\varepsilon)\mathbb{E}_s[R_s]}{(1-\varepsilon)\mathbb{E}_s[D_s]}, \frac{(1-\varepsilon)\mathbb{E}_s[R_s]}{(1+\varepsilon)\mathbb{E}_s[D_s]}\right]$$

Finally, if $\mathbb{E}_s[R_s] = 0$, we have that

$$\mathbb{E}_s[MPR \mid G] \in \left[-\frac{\varepsilon}{(1-\varepsilon)\mathbb{E}_s[D_s]}, \frac{\varepsilon}{(1-\varepsilon)\mathbb{E}_s[D_s]}\right]$$

In any case, since this is true for any $\varepsilon > 0$, we have that $\mathbb{E}_s[MPR] = \frac{\mathbb{E}_s[R_s]}{\mathbb{E}_s[D_s]}$, as wanted. □

Definition A.3. Let $M = (V, p, r, d)$ and $M' = (V', p', r', d')$ be two Markov chains with associated rewards and durations, where $V, V' \subseteq \mathbb{N}_{\leq n}$. We call M' a *summary* M if

(1) There is a one-to-one correspondence between CCSs $C_0, C_1, \ldots$ in M and and $C'_0, C'_1, \ldots$ in M' that agrees on states in $V \cap V'$. That is, for any pair C_j and C'_j of corresponding CCSs, it holds that $C_j \cap V' = C'_j \cap V$.
(2) The probability of reaching any a CCS C_j in M is the same as reaching the corresponding C'_j in M'.
That is, $\mathbb{P}^M_s\left(H_{C_j} < \infty\right) = \mathbb{P}^{M'}_s\left(H_{C'_j} < \infty\right)$ for any $s \in V \cap V'$.
(3) The expected cumulative rewards and durations on cycles from and to recurrent states are preserved.
That is, for all s recurrent, $\mathbb{E}^M_s[R_s] = \mathbb{E}^{M'}_s[R_s]$ and $\mathbb{E}^M_s[D_s] = \mathbb{E}^{M'}_s[D_s]$.

The following lemma shows that summaries preserve the expected mean-payoff ratio values.

Lemma A.4. *If M' is a summary of M and $s \in V \cap V'$. Then $\mathbb{E}^M_s[MPR] = \mathbb{E}^{M'}_s[MPR]$.*

Proof. Let $B \subseteq V$ be a maximal set of recurrent states that do not communicate pairwise. That is, B consists of representative states, one for each CCS of M. Then

$$\mathbb{E}^M_s[MPR] = \sum_{b \in B} \mathbb{P}^M_s(H_b < \infty) \cdot \mathbb{E}^M_b[MPR]\,. \tag{2}$$

This holds because in finite Markov chains the absorption probability is 1, i.e., almost-surely eventually a state in B will be reached, and that the *MPR* objective is prefix-independent. Now from the assumption that M' is a summary of M we get the following. Notice that a state $b \in B$ represents a CCS both in M and M' by point 1 of Definition A.3.

$$\begin{aligned}
\mathbb{E}_s^M[MPR] &= \sum_{b\in B} \mathbb{P}_s^M(H_b < \infty) \cdot \mathbb{E}_b^M[MPR] && \text{Equation (2)}\\
&= \sum_{b\in B} \mathbb{P}_s^M(H_b < \infty) \cdot \frac{\mathbb{E}_b^M[R_b]}{\mathbb{E}_b^M[R_b]} && \text{Lemma A.2}\\
&= \sum_{b\in B} \mathbb{P}_s^{M'}(H_b < \infty) \cdot \frac{\mathbb{E}_b^{M'}[R_b]}{\mathbb{E}_b^{M'}[R_b]} && \text{Definition A.3}\\
&= \sum_{b\in B} \mathbb{P}_s^{M'}(H_b < \infty) \cdot \mathbb{E}_b^M[MPR] = \mathbb{E}_s^{M'}[MPR] && \text{Equation (2).}
\end{aligned}$$

□

It remains to show that the operations of eliminating loops and states create summaries.

Lemma A.5 (Edge Collapse). *Let M be a Markov chain, $i, j \in V$ two states with $p(i,j) = 0$ and $S \subseteq V$ be a set of intermediate states x all satisfying $p(x,j) = 1$ and $p(k,x) = 0$ for all $k \neq i$. Let M' be the Markov chain obtained by simultaneously replacing all length-2-paths from i to j via S by just one direct edge with the same expected reward and duration. That is, in M' we have that*

- $p'(i,j) = \sum_{x\in S} p(i,x)$
- $r'(i,j) = \sum_{x\in S} p(i,x)(r(x,i) + r(x,j))$
- $d'(i,j) = \sum_{x\in S} p(i,x)(d(x,i) + d(x,j))$

Then M' is a summary of M.

Proof. For condition (1) in Definition A.3, notice that any two states $s, t \in V$ communicate in M iff they do in M'.

For point (2), recall that the probabilities $R : V \to \mathbb{R}$ of reaching a set $C \subseteq V \setminus S$ of states are the least fixed-point satisfying

$$R(s) = \begin{cases} 1 & \text{if } s \in C \\ \inf\{\sum_{k\in V} p(s,k)R(k)\} & \text{otherwise} \end{cases}$$

By definition of M' the sum in the second case can be rewritten as

$$\begin{aligned}
\sum_{k\in V} p(i,k)R(k) &= \sum_{\substack{s\neq i\\ k\in V}} p(s,k)R(k) + \sum_{\substack{s=i\\ k\in V\setminus S}} p(i,k)R(k) + \sum_{\substack{s=i\\ k\in S}} p(i,k)R(k)\\
&= \sum_{\substack{s\neq i\\ k\in V}} p'(s,k)R(k) + \sum_{\substack{s=i\\ k\in V\setminus S}} p'(i,k)R(k) + p'(i,j)R(j) = \sum_{k\in S} p'(s,k)R(k)
\end{aligned}$$

We conclude that a least solution of the system of equations is the same for M and M', which implies the claim.

For (3), we show the claim for rewards only; the proof that expected total durations on cycles is analogous. Pick a recurrent state s and consider the expectation Clearly, if state i is not part of the same CCS as s then $\mathbb{E}_s^M[R_s] = \mathbb{E}_s^{M'}[R_s]$. So suppose now that s, i and j are part of the same CCS.

Note that by assumption on M, for every state $k \neq i$ in the same CCS it holds that $\mathbb{E}_k^M[H_i] < \mathbb{E}_k^M[H_S]$ and therefore

$$\mathbb{E}_k^M[R_i] = \mathbb{E}_k^{M'}[R_i] \tag{3}$$

We can split $\mathbb{E}_i^M[R_i]$ into two parts, according to the disjoint cases or not a state of S is visited in the first step.

The first case is

$$\mathbb{E}_i^M[R_i \mid H_S = 1] = \sum_{k\in S} p(i,k)\, r(i,k) + \mathbb{E}_k^M[R_i] = p'(i,j) + \mathbb{E}_j^M[R_i] = \mathbb{E}_i^{M'}\left[R_i \mid H_j = 1\right] \tag{4}$$

The second case is

$$\mathbb{E}_i^M[R_i \mid H_S > 1] = \sum_{k\notin S} p(i,k)\, r(i,k) + \mathbb{E}_k^M[R_i] = \sum_{k\notin S} p'(i,k)\, r(i,k) + \mathbb{E}_k^{M'}[R_i] = \mathbb{E}_i^{M'}\left[R_i \mid H_j > 1\right] \tag{5}$$

Using that the two events are disjoint we get

$$\begin{aligned}
\mathbb{E}_i^M[R_i] &= \mathbb{E}_i^M[R_i \mid H_S = 1] + \mathbb{E}_i^M[R_i \mid H_S > 1]\\
&= \mathbb{E}_i^{M'}\left[R_i \mid H_j = 1\right] + \mathbb{E}_i^{M'}\left[R_i \mid H_j > 1\right] = \mathbb{E}_i^{M'}[R_i]
\end{aligned} \tag{6}$$

For arbitrary s we can write

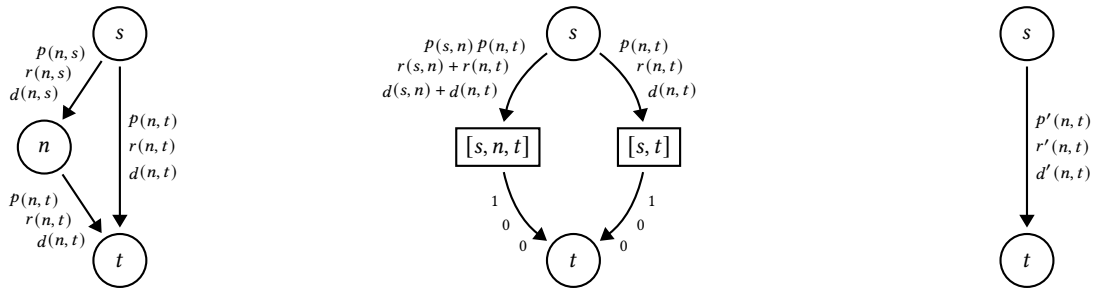


Figure 4: State elimination: On the left is (part of a) Markov chain M before removing state n and on the right is the corresponding part of Markov chain M'. In the middle is the intermediate step M'' as constructed in the proof of Lemma 5.2. The probability, and expected duration and reward, of moving from state i to j remains untouched.

$$\mathbb{E}_s^M[R_s] = \sum_{n \geq 0} \mathbb{E}_s^M\left[R_i \mid N_t^i = n\right] \tag{7}$$

where N_t^i denotes the number of times state i is visited between time 1 and H_t. Clearly the expected reward sum is the same in M and M' on paths that do not visit state i.That is, $\mathbb{E}_s^M\left[R_t \mid N_t^i = 0\right] = \mathbb{E}_s^{M'}\left[R_t \mid N_t^i = 0\right]$. Now each summand on the RHS of Equation (7) can be written as

$$\begin{aligned}
\mathbb{E}_s^M\left[R_i \mid N_t^i = n\right] &= \mathbb{E}_s^M\left[R_i \mid N_t^i = 0\right] + \mathbb{E}_i^M\left[R_i \mid N_t^i = n\right] + \mathbb{E}_i^M\left[R_t \mid N_t^i = 0\right] \\
&= \mathbb{E}_s^M\left[R_i \mid N_t^i = 0\right] + n \cdot \mathbb{E}_i^M[R_i] + \mathbb{E}_i^M\left[R_t \mid N_t^i = 0\right] \\
&= \mathbb{E}_s^{M'}\left[R_i \mid N_t^i = 0\right] + n \cdot \mathbb{E}_i^{M'}[R_i] + \mathbb{E}_i^{M'}\left[R_t \mid N_t^i = 0\right] \\
&= \mathbb{E}_s^{M'}\left[R_t \mid N_t^i = n\right]
\end{aligned}$$

where the second equality uses the Markov property and the third one is due to Equations (6) and (7). This concludes the proof that M' is a summary of M. □

Proof of Lemma 5.2. To show that eliminating states preserves the mean-payoff ratio values we split the transformation from M to M' in two steps via an intermediate Markov chain M'' and argue that M' is a summary of M'' which in turn is a summary of M. The claim then follows from Lemma A.4.

We define M'' from M by introducing new intermediate states $[s, n, t]$ and $[s, t]$ between any two states $s, t \in V$ of M, replacing length-two paths from s to t via n by paths via $[s, n, t]$, and length-one paths from s to t by paths via $[s, t]$. See Figure 4 for an illustration. Formally,

$$p''(s, n) = 0; \quad p''(s, [s, n, t]) = p(s, n)\, p(n, t); \quad p''([s, n, t], t) = 1; \text{ and } \quad p''(k, [s, n, t]) = 0 \text{ for all } k \neq s.$$

The rewards and durations incurred on those steps are

$$\begin{aligned}
r''(s, [s, n, t]) &= r(s, t) + r(n, t); \\
d''(s, [s, n, t]) &= d(s, t) + d(n, t); \quad \text{and} \\
r''([s, n, t], t) &= d''([s, n, t], t) = 0.
\end{aligned}$$

Any path from s to t via the new state $[s, t]$ in M' directly corresponds to a length-one path $s \longrightarrow t$ in M'.

$$p''(s, [s, t]) = p(s, t); \text{ and } \quad p''([s, t], t) = 1$$

The rewards and durations incurred on those steps are

$$r''(s, [s, t]) = r(s, t); \quad d''(s, [s, t]) = d(s, t) \text{ and } \quad r''([s, t], t) = d''([s, t], t) = 0.$$

Notice that for any fixed pair $i, j \neq n$ of states, $S = \{[i, j], [i, n, j]\}$ satisfies the assumption of Lemma A.5 and M' is result of collapsing them accordingly. So by multiple applications of Lemma A.5 we observe that M' is a summary of M''. It remains to show that M'' is a summary of M.

For (1), just notice that going from M to M'' does not change which states in $s, t \in V$ communicate: they do in M iff they do in M''. The newly introduced states $[s, t]$ and $[s, n, t]$ are in a CCS C iff $s, n \in C$.

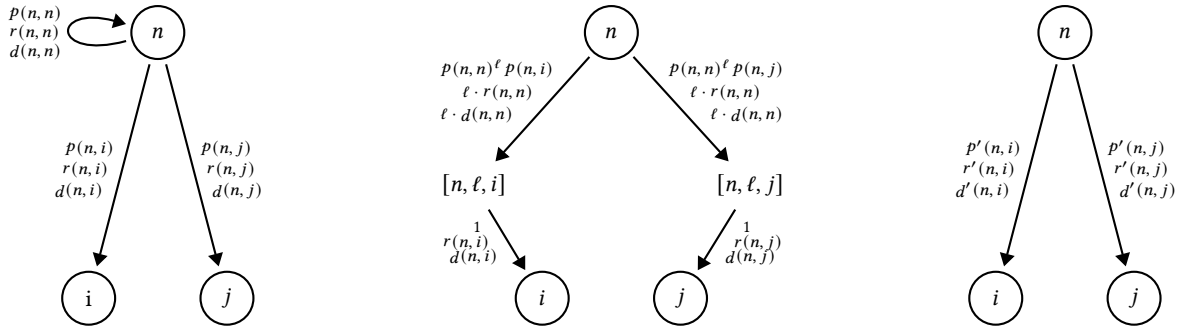


Figure 5: Loop elimination: On the left is (part of the) Markov chain M before eliminating the self-loop in vertex n; On the right is the resulting chain M'. In the middle is the intermediate chain M'' with the countably infinite edges between n and auxiliary state n'. Taking the ℓth edge represents taking the loop ℓ times.

For (2), first recall that reaching a set $C \subseteq V$ is an open objective and thus can be expressed as the countable sum

$$\mathbb{P}_s^M(H_C < \infty) = \sum_{\substack{\pi = s_0 s_1 \dots s_k \in V^* \\ s_0 = s \wedge s_k \in C}} p(\pi) \tag{8}$$

where $p(\pi) = \prod_{i=1}^{k} p(s_{i-1}, s_i)$ is the product of probabilities along π. We observe that there is an obvious isomorphism $f : V^* \to V(V'')^*V$ between finite paths in M that lead from s to C, and those in M'' that lead from s to C, which preserves probability mass: simply let $f(inj) = i[i, n, j]j$ and $f(ij) = i[ij]j$. Then $p(\pi) = p''(f(\pi))$. We can decompose $\mathbb{P}_s^{M''}(H_C < \infty)$ as in Equation (8) but since neither source nor target consists of states in $V'' \setminus V$ we have that

$$\mathbb{P}_s^M(H_C < \infty) = \sum_{\substack{\pi = s_0 s_1 \dots s_k \in V^* \\ s_0 = s \wedge s_k \in C}} p(\pi) = \sum_{\substack{\pi = s_0 s_1 \dots s_k \in V^* \\ s_0 = s \wedge s_k \in C}} p''(f(\pi)) = \sum_{\substack{\pi'' = s_0 s_1 \dots s_k \in V(V'')^*V \\ s_0 = s \wedge s_k \in C}} p''(\pi'') = \mathbb{P}_s^{M''}(H_C < \infty).$$

Point (3) follows analogously to point (2) above. We have that

$$\mathbb{E}_s^M[R_s] = \sum_{\substack{\pi = s_0 s_1 \dots s_k \in V^* \\ s_0 = s = s_k}} p(\pi) \cdot r(\pi) = \sum_{\substack{\pi = s_0 s_1 \dots s_k \in V^* \\ s_0 = s = s_k}} p''(f(\pi)) \cdot r''(f(\pi)) = \mathbb{E}_s^{M''}[R_s]$$

where $r(\pi) = \sum_{i=1}^{|\pi|} r(s_{i-1}, s_i)$ is the total reward along path π. □

Proof of Lemma 5.4. Consider the Markov chain M and let n be the state whose loop is eliminated. We define an intermediate Markov chain M'' that replaces the self-loop by a countably infinite number of new states, each of which represents a fixed number of iterations of the loop.

That is M'' is obtained from M as follows. For every $\ell \geq 0$ and $j \neq n$ there is a new state $[n, \ell, j]$. We let

$$p''(n, [n, \ell, j]) \stackrel{\text{def}}{=} p(n, n)^\ell \, p(n, j)$$
$$r''(n, [n, \ell, j]) \stackrel{\text{def}}{=} \ell \cdot r(n, n)$$
$$d(n, [n, \ell, j]) \stackrel{\text{def}}{=} \ell \cdot d(n, n)$$

and $p''([n, \ell, j], j) \stackrel{\text{def}}{=} 1$ and $r''([n, \ell, j], j) \stackrel{\text{def}}{=} d''([n, \ell, j], j) \stackrel{\text{def}}{=} 0$.

Every original edge between states $i, j \neq n$ has the same probability, reward and duration as in M. See also Figure 5 in the middle.

Intuitively, M'' is the (countably infinite) unfolding of the self-loop around state n. Every path in M that goes from n to j has some fixed number $\ell \geq 0$ of iterations of the loop and corresponds to a path $n \to [n, \ell, j] \to j$ in M'' which has the same probability, reward and duration. Note in particular that the probability of a path $n \to \dots \to n \to j$ that iterates the loop exactly ℓ times and then goes to j is $p(n, n)^\ell (1 - p(n, n)) \cdot \frac{p(n,j)}{1-p(n,n)} = p''(n, [n, \ell, j])$. One thus readily verifies that M'' is a summary of M in the sense of Definition A.3.

It remains to observe that M' is a summary of M''. This is because it is the result of removing intermediate states via Lemma A.5. We conclude that M' is a summary of M. The claim of the lemma now follows from Lemma A.4. □

B PROOFS FOR SECTION 6

Lemma 6.2. *For any k-player game G, and a player i, one can construct a formula $Val_i(\mathbf{x}_1, \ldots, \mathbf{x}_k, y)$, such that $\mathbf{x}_j$ encodes strategy σ_j of a strategy profile $\sigma \in \mathcal{B}(\mathbf{b})$, and the formula uses a set of Boolean combination of $100N^3 + 40N$ polynomials with at most $8N^3 + 3N$ variables, each of degree at most $2k$, and coefficients coming from G and y encode the outcome of σ for player i.*

In this section, we show how to find the outcome of an induced Markov chain for a specific player in state 1 in the first order theory over the reals. The problem, in particular, is that depending on which strategy profile we picked, the induced Markov chain could have different (end) strongly connected components and we need the formula to be the same no matter which strategy profile we picked. Still, our solution is to implement loop and state elimination, see Section 5, while taking care that some of these steps might not be possible to do (for example, you cannot do state elimination of a state that is absorbing with incoming edges or if it has already been eliminated, and you cannot do loop elimination on a state if there is no other successor of the state or if it has been eliminated).

For each pair of (1) memory vectors $\mathbf{m}$ and $\mathbf{m}'$, (2) game states v and v' and (3) signal vectors $\mathbf{s}$ and $\mathbf{s}'$ and for each iteration i (an iteration consists of either one loop or one state elimination step), we have variables corresponding to the (possible) state (v, m, s) and edge $((v, m, s), (v', m', s'))$ of the induced Markov chain, describing the Markov chain at the beginning of iteration i. For simplicity, we will talk about (v, m, s) as being a state of the Markov chain. Therefore, there are at most N states and we only need to assign value to the states of the Markov chain initially.

To be explicit about it, we first run a loop and then state elimination on state x for each $x \in \{N, N-1, \ldots, 1\}$, except we do not do state elimination on state 1. In other words, for each $1 \leq j < 2N$, in iteration j for odd j, we do loop elimination on state $N + 1 - (j+1)/2$ and in iteration j for even j, we do state elimination on state $N + 1 - j/2$.

Specifically, we have variables, each describing the Markov chain at the start of iteration j. Let $S = (v, \mathbf{m}, \mathbf{s})$ and $T = (v', \mathbf{m}', \mathbf{s}')$. The intention of them is as follows:

(1) $x_{S,j}$ should be 1 if state s exists and otherwise 0.
(2) $x_{S,T,j}$ should be 1 if action (S, T) exists and otherwise 0.
(3) $q_{S,T,j}$ should be the probability to use action (S, T) when in S.
(4) $n_{S,T,j}$ should be the duration of action (S, T).
(5) $r_{S,T,j}$ should be the reward of action (S, T) for the specific player.

We have some expressions (i.e. some boolean combinations of polynomials) giving the value of each of these variables at the start of iteration 1 and depending on the value of the variables at the start of a loop or state elimination iteration, we have expressions giving the value of each of these variables after that iteration. The conjunction of all of these expressions give the full formula describing the outcome of the game for the specific player.

Setting the initial values of the variables. Note that expressions such as

$$\sum_{\mathbf{a}=(a_1,\ldots,a_k)|\Delta(v,\mathbf{a})(*,s',v')>0} \prod_{i=1}^{k} q_{a_i}^{s_i,m_i,i} b_{m_i'}^{s_i',m_i,i}$$

are explicit polynomials given the game and states (v, m, s) and (v', m', s') explicitly because the set of such a's comes directly from the input game ($\Delta(v, a)(*, s', v')$ is the probability to go from v to v' when the players plays a, emitting signals s', ignoring the reward) and for a given a the product is explicit. The expressions at the beginning, i.e. for iteration 1, are as follows:

(1) The expression for $x_{(v,\mathbf{m},\mathbf{s}),1}$ is
$$x_{(v,\mathbf{m},\mathbf{s}),1} = 1$$
(one could omit these variables, but including them makes it easier to follow). That is, initially, all states exists.
(2) The expression for $x_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1}$, for $\mathbf{m} = (m_1, m_2, \ldots, m_k)$, $\mathbf{s} = (s_1, s_2, \ldots, s_k)$, $\mathbf{m}' = (m_1', m_2', \ldots, m_k')$ and $\mathbf{s}' = (s_1', s_2', \ldots, s_k')$ is
$$(x_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1} = 1 \wedge \sum_{a=(a_1,\ldots,a_k)|\Delta(v,a)(*,s',v')>0} \prod_{i=1}^{k} q_{a_i}^{s_i,m_i} \cdot b_{m_i'}^{s_i',m_i,i} > 0)$$
$$\vee (x_{(v,\mathbf{m},\mathbf{m}),(v',\mathbf{m}',\mathbf{s}'),1} = 0 \wedge \sum_{a=(a_1,\ldots,a_k)|\Delta(v,a)(*,s',v')>0} \prod_{i=1}^{k} q_{a_i}^{s_i,m_i,i} \cdot b_{m_i'}^{s_i',m_i,i} = 0)$$
In words, the expression sets $x_{(v,m,s),(v',m',s'),1}$ to 1 iff there is an action a that happens with positive probability which has v' while emitting $\mathbf{s}'$ as an outcome, and the probability to update from $\mathbf{m}$ to $\mathbf{m}'$ on signal $\mathbf{s}$ (i.e. for each player, the probability to update from m_i to m_i' on signal s_i) is also positive.
(3) The expression for $q_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1}$ is
$$q_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1} = \sum_{a=(a_1,\ldots,a_k)|\Delta(v,a)(*,\mathbf{s}',v')>0} \prod_{i=1}^{k} q_{a_i}^{s_i,m_i,i} \cdot b_{m_i'}^{s_i',m_i,i} \cdot \Delta(v,a)(*,s',v') \ .$$

In words, the expression $\prod_{i=1}^{k} q_{a_i}^{s_i,m_i,i}$ is the probability of using action profile a when the signals and memory were $\mathbf{s}$ and $\mathbf{m}$ respectively. That multiplied by $\prod_{i=1}^{k} b_{m_i'}^{s_i,m_i,a,i}$ is then the probability to update $\mathbf{m}$ to $\mathbf{m'}$ as well, on signal $\mathbf{s}$. Finally, that times $\Delta(v,a)(*,s',v')$ is then the probability to go to v', and emitting signal $\mathbf{s'}$ to the players, when in v with the signal being s, and the players used action a. The full expression is then summing over all possible action profiles a.

(4) The expression for $n_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1}$ is

$$n_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1} = 1 \ .$$

I.e. the length of the edge initially is 1.

(5) The expression for $r_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}),1}$ is

$$r_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1} q_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1} = \sum_{r|r \text{ is a reward for the selected player } a=(a_1,\ldots,a_n)|\Delta(v,a)(r,\mathbf{s}',v')>0} \prod_{i=1}^{k} q_{a_i}^{s_i,m_i,i} b_{m_i'}^{s_i',m_i,i} \Delta(v,a)(r,\mathbf{m}',\mathbf{s}')r \ .$$

In words, $\prod_{i=1}^{k} q_{a_i}^{s_i,m_i,i} b_{m_i'}^{s_i',m_i,i} \Delta(v,a)(r,\mathbf{m}',s')$ expresses the probability that, when in state v and emitting signal $\mathbf{s}$ and the players have memory $\mathbf{m}$, we go to v' emitting s' and the players update memory to $\mathbf{m'}$, while the players used joint action a and the specific player we focus on gets reward r. We then sum over all possible r's and a's to get the expected reward. The reason we have $r_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1} q_{(v,\mathbf{m},\mathbf{s}),(v',\mathbf{m}',\mathbf{s}'),1}$ on the left-hand side is that rewards in a Markov chain for an edge are conditional on following that edge (it is perhaps easiest to understand using an example: If the game has only a single action a, memory m and signal s for each player in every state and we have that $\Delta(v,a)$, for some state v, is the uniform distribution over $(1,s,t)$ and $(1,s,u)$ for some states t,u, then the right hand side of the expression for going from v to t is $1/2$ because $\Delta(v,a)(1,s,t) = 1/2$ and each other variable in it is 1 (or 0). But clearly, the reward for going from v to t should be 1, because that is the reward for each action in the game!).

For simplicity, in the remainder, we write S for $(v,\mathbf{m},\mathbf{s})$ and T for $(v',\mathbf{m}',\mathbf{s}')$, because these are the states of the Markov chain.

We will next give the expressions for each variable and each $1 < j < 2N$, depending on whether j is odd or even (i.e. depending on whether we are doing loop or state elimination).

Loop elimination. For even j (i.e. we are doing loop elimination from $j-1$ to j on some state $v = N+1-j/2$), we use the following expressions:

(1) The expression for $x_{S,j}$ is

$$x_{S,j} = x_{S,j-1} \ .$$

In words, each state exists if it did before.

(2) The expression for $x_{v,v,j}$ is

$$(x_{v,v,j} = 0 \wedge \sum_{S|S\neq v} x_{v,s,j-1} > 0) \vee (x_{v,v,j} = 1 \wedge \sum_{S|S\neq v} x_{v,s,j-1} = 0) \ .$$

In words, the loop (v,v) should not exist after this iteration if v has another successor (the sum $\sum_{S|S\neq v} x_{v,S,j-1}$ is 0 precisely if for all $S \neq v$, (v,S) does not exist). We can not eliminate a loop if the state does not have another successor though.
For each S,T, except $S = T = v$, the expression for $x_{S,T,j}$ is

$$x_{S,T,j} = x_{S,T,j-1} \ .$$

In words, the edge is still there if it was before.

(3) The expression for $q_{v,S,j}$, for each $S \neq v$ is

$$(x_{v,v,j-1} = 1 \wedge x_{v,v,j} = 0 \wedge q_{v,S,j}(1 - q_{v,v,j-1}) = q_{v,S,j-1}) \vee$$
$$(x_{v,v,j-1} = 0 \wedge x_{v,v,j} = 0 \wedge q_{v,S,j} = q_{v,S,j-1})$$

In words, if we are doing loop elimination (i.e. the self-loop existed before but not after $x_{v,v,j-1} = 1 \wedge x_{v,v,j} = 0$), then, $q_{v,S,j} = q_{v,S,j-1}/(1 - q_{v,v,j-1}) \Rightarrow q_{v,S,j}(1 - q_{v,v,j-1}) = q_{v,S,j-1}$), because $q_{v,v,j-1} \neq 1$ - if the probability of using the self-loop had been 1, we had no other successors of v than v and thus we could not do loop elimination. The other part is saying that if there were no self-loop before and after, then the value of $q_{v,S,j}$ is as before. This is fine to do even if the edge (v,S) does not exist, but one could also test that similarly. Note that we do not consider the case where there is a self-loop both before and after even if it can occur: It only occurs in case there are no $S \neq v$ such that (v,S) exists and in that case, we do not need to set $q_{v,S,j}$ to anything in particular.
The expression for $q_{S,T,j}$ for each S,T such that $S \neq v$ is

$$q_{S,T,j} = q_{S,T,j-1} \ .$$

In words, the value of the edge probability is as before. Again, this is fine to do even if the edge (v,S) does not exist, but one could also test that similarly.
Note that we are not setting $q_{v,v,j}$ at all, since its value does not matter as long as the edge does not exist.

(4) The expression for $n_{v,S,j}$, for each $S \neq v$ is

$$(x_{v,v,j-1} = 1 \wedge x_{v,v,j} = 0 \wedge n_{v,S,j}(1 - q_{v,v,j-1}) = q_{v,v,j-1}n_{v,v,j-1} + n_{v,S,j-1}(1 - q_{v,v,j-1})) \vee$$
$$(x_{v,v,j-1} = 0 \wedge x_{v,v,j} = 0 \wedge n_{v,S,j} = n_{v,S,j-1}) \ .$$

In words, if we are doing loop elimination (i.e. the self-loop existed before but not after $x_{v,v,j-1} = 1 \wedge x_{v,v,j} = 0$), then, since we took the loop $q_{v,v,j-1}/(1 - q_{v,v,j-1})$ many times in expectation, each taking $n_{v,v,j-1}$ many steps, we need to add that in, giving us $n_{v,S,j} = q_{v,v,j-1}n_{v,v,j-1}/(1 - q_{v,v,j-1}) + n_{v,S,j} \Rightarrow n_{v,S,j}(1 - q_{v,v,j-1}) = q_{v,v,j-1}n_{v,v,j-1} + n_{v,S,j}(1 - q_{v,v,j-1})$. On the other hand, if we did not have a loop to eliminate, the number of steps for this edge is as before. Finally, like in the previous case, if we did not do loop elimination or if this edge does not exist, we can set the value of $n_{v,S,j}$ arbitrarily.
The expression for $n_{S,T,j}$, for each $S \neq v$ is

$$n_{S,T,j} = n_{S,T,j-1} \ .$$

In words, we do not change the edge at all if it is not going out of v.

(5) The expression for $r_{v,S,j}$, for each $S \neq v$ is

$$(x_{v,v,j-1} = 1 \wedge x_{v,v,j} = 0 \wedge r_{v,S,j}(1 - q_{v,v,j-1}) = q_{v,v,j-1}r_{v,v,j-1} + r_{v,S,j-1}(1 - q_{v,v,j-1})) \vee$$
$$(x_{v,v,j-1} = 0 \wedge x_{v,v,j} = 0 \wedge r_{v,S,j} = r_{v,S,j-1}) \ .$$

In words, this is analogous to how we updated $n_{v,S,j}$
The expression for $r_{S,T,j}$, for each $S \neq v$ is

$$r_{S,T,j} = r_{S,T,j-1} \ .$$

In words, we do not change the edge at all if it is not going out of v.

State elimination. For odd $j > 3$ (i.e. we are doing state elimination from $j-1$ to j on some state $v = N + 1 - (j-1)/2$), we use the following expressions:

(1) The expression for $x_{v,j}$ is

$$(x_{v,j} = 0 \wedge x_{v,j-1} = 0) \vee$$
$$(x_{v,j} = 1 \wedge x_{v,j-1} = 1 \wedge \sum_{S \neq v} x_{S,v,j-1} > 0 \wedge \sum_{S \neq v} x_{v,S,j-1} = 0) \vee$$
$$(x_{v,j} = 0 \wedge x_{v,j-1} = 1 \wedge (\sum_{S \neq v} x_{S,v,j-1} = 0 \vee \sum_{S \neq v} x_{v,S,j-1} > 0))$$

In words, we can not eliminate v if it has already been eliminated, or if it is absorbing and have incoming edges. Note that $\sum_{S \neq v} x_{S,v,j-1}$ is the number of incoming edges to v and $\sum_{S \neq v} x_{v,S,j-1}$ is the number of outgoing from v, in both cases ignoring loops.
For each other state $S \neq v$ the expression for $x_{S,j}$ is

$$x_{S,j} = x_{S,j-1}$$

In words, it exists if it did before.

(2) The expression for $x_{v,S,j}$, for each $S \neq v$ is as follows:

$$(x_{v,j} = 1 \wedge x_{v,S,j} = x_{v,S,j-1}) \vee (x_{v,j} = 0 \wedge x_{v,S,j} = 0)$$

In words, if we did not do state elimination, we do nothing to the edge, otherwise, we remove it.
Similarly, the expression for $x_{S,v,j}$, for each $S \neq v$ is as follows:

$$(x_{v,j} = 1 \wedge x_{S,v,j} = x_{S,v,j-1}) \vee (x_{v,j} = 0 \wedge x_{S,v,j} = 0)$$

Also, the expression for $x_{v,v,j}$ is as follows:

$$(x_{v,j} = 1 \wedge x_{v,v,j} = x_{v,v,j-1}) \vee (x_{v,j} = 0 \wedge x_{v,v,j} = 0)$$

Finally, the expression for each other edge $x_{S,T}$ for $S \neq v \neq T$, is as follows:

$$(x_{v,j} = 1 \wedge x_{S,T,j} = x_{S,T,j-1}) \vee$$
$$(x_{v,j} = 0 \wedge x_{S,T,j} = 1 \wedge (x_{S,T,j-1} = 1 \vee x_{S,v,j-1} + x_{v,T,j-1} = 2)) \vee$$
$$(x_{v,j} = 0 \wedge x_{S,T,j} = 0 \wedge x_{S,T,j-1} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} < 2)$$

In words, if we did not do state elimination we did nothing, otherwise, there is an edge (S,T) if there were one before or if there were an edge from S to v and one from v to T and otherwise not.

(3) The expression for $q_{S,T,j}$, for each $S \neq v \neq T$ is

$$\begin{aligned}
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} = 2 \wedge x_{S,T,j-1} = 1 \wedge q_{S,T,j} = q_{S,T,j-1} + q_{S,v,j-1} * q_{v,T,j-1}) \vee \\
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} < 2 \wedge q_{S,T,j} = q_{S,T,j-1}) \vee \\
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} = 2 \wedge x_{S,T,j-1} = 0 \wedge q_{S,T,j} = q_{S,v,j-1} * q_{v,T,j-1}) \vee \\
&(x_{v,v,j} = 1 \wedge q_{S,T,j} = q_{S,T,j-1})
\end{aligned}$$

In words, the first clause is saying that if we do state elimination, we had edges (S, v), (v, T) and (S, T), then the probability to go through (S, T) after the elimination is the probability of going from S to v to T plus the probability to go (S, T) from before we did state elimination. The second clause is saying that if we do state elimination and had (S, T) but not both (S, v) and (v, T) then the probability for (S, T) is as before. The third is saying that if we do state elimination and had both (S, v) and (v, T), but not (S, T) then the probability for (S, T) after is the probability of going S to v to T before. The last is saying that if we did not do state elimination, then the probability remains unchanged.
The expression for $q_{v,S,j}$, for each S is

$$(x_{v,v,j} = 1 \wedge q_{v,S,j} = q_{v,S,j-1}) \vee x_{v,v,j} = 0$$

In words, if we did not do state elimination, then the probability remains the same. If we did state elimination we do not care about $q_{v,S,j}$. We could as such equally have used the expression $q_{v,S,j} = q_{v,S,j-1}$, since we do not care about the value of $q_{v,S,j}$ if we did do state elimination, but this makes the construction easier to follow.
The expression for $q_{S,v,j}$, for each $S \neq v$ is

$$(x_{v,v,j} = 1 \wedge q_{S,v,j} = q_{S,v,j-1}) \vee x_{v,v,j} = 0$$

In words, if we did not do state elimination, then the probability remains the same. If we did state elimination we do not care about $q_{S,v,j}$. Like above, we could do $q_{S,v,j} = q_{S,v,j-1}$ instead.

(4) The expression for $n_{S,T,j}$, for each $S \neq v \neq T$ is

$$\begin{aligned}
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} = 2 \wedge x_{S,T,j-1} = 1 \wedge \\
&n_{S,T,i}(q_{S,T,j-1} + q_{S,v,j-1}q_{v,T,j-1}) = n_{S,T,j-1}q_{S,T,j-1} + q_{S,v,j-1}q_{v,T,j-1}(n_{S,v,j-1} + n_{v,T,j-1})) \vee \\
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} < 2 \wedge n_{S,T,j} = n_{S,T,j-1}) \vee \\
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} = 2 \wedge x_{S,T,j-1} = 0 \wedge n_{S,T,j} = n_{S,v,j-1} + n_{v,S,j-1}) \vee \\
&(x_{v,v,j} = 1 \wedge n_{S,T,j} = n_{S,T,j-1})
\end{aligned}$$

In words, the first clause (which is over the first two lines) is saying that if we do state elimination, we had edges (S, v), (v, T) and (S, T), then the expected length to go through (S, T) after the elimination is

$$\frac{n_{S,T,j-1}q_{S,T,j-1}}{q_{S,T,j-1} + q_{S,v,j-1}q_{v,T,j-1}} + \frac{q_{S,v,j-1}q_{v,T,j-1}(n_{S,v,j-1} + n_{v,T,j-1})}{q_{S,T,j-1} + q_{S,v,j-1}q_{v,S,j-1}},$$

i.e. if we went through (S, T) after, it corresponded to going from S to T directly before or from S to v to T and those options are not necessarily equally likely, so we need to multiply with the conditional probability of going through those edges. The second clause is saying that if we do state elimination and had (S, T) but not both (S, v) and (v, T) then the length of (S, T) is as before. The third is saying that if we do state elimination and had both (S, v) and (v, T), but not (S, T) then the length of (S, T) after is the length of the path from S to v to T. The last is saying that if we did not do state elimination, then the length remains unchanged.
The expression for $n_{v,S,j}$, for each S is

$$(x_{v,v,j} = 1 \wedge n_{v,S,j} = n_{v,S,j-1}) \vee x_{v,v,j} = 0$$

In words, if we did not do state elimination, then the length remains the same. If we did state elimination we do not care about $n_{v,S,j}$. We could as such equally have used the expression $n_{v,S,j} = n_{v,S,j-1}$, since we do not care about the value of $n_{v,S,j}$ if we did do state elimination, but like for the probability it seemed harder to follow.
The expression for $n_{S,v,j}$, for each $S \neq v$ is

$$(x_{v,v,j} = 1 \wedge n_{S,v,j} = n_{S,v,j-1}) \vee x_{v,v,j} = 0$$

In words, if we did not do state elimination, then the length remains the same. If we did state elimination we do not care about $n_{S,v,j}$. Like above, we could do $n_{S,v,j} = n_{S,v,j-1}$ instead.

(5) The expression for $r_{S,T,j}$ is much the same as for $n_{S,T,j}$ but is included for completeness.

The expression for $r_{S,T,j}$, for each $S \neq v \neq T$ is

$$\begin{aligned}
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} = 2 \wedge x_{S,T,j-1} = 1 \wedge \\
&r_{S,T,j}(q_{S,T,j-1} + q_{S,v,j-1}q_{v,S,j-1}) = r_{S,T,j-1}q_{S,T,j-1} + q_{S,v,j-1}q_{v,T,j-1}(r_{S,v,j-1} + r_{v,T,j-1})) \vee \\
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} < 2 \wedge r_{S,T,j} = r_{S,T,j-1}) \vee \\
&(x_{v,v,j} = 0 \wedge x_{S,v,j-1} + x_{v,T,j-1} = 2 \wedge x_{S,T,j-1} = 0 \wedge r_{S,T,j} = r_{S,v,j-1} + r_{v,T,j-1}) \vee \\
&(x_{v,v,j} = 1 \wedge r_{S,T,j} = r_{S,T,j-1})
\end{aligned}$$

In words, the first clause (which covers the first two lines) is saying that if we do state elimination, we had edges (S, v), (v, T) and (S, T), then the expected reward to go through (S, T) after the elimination is $r_{S,T,j-1}q_{S,T,j-1}/(q_{S,T,j-1} + q_{S,v,j-1}q_{v,T,j-1}) + q_{S,v,j-1}q_{v,T,j-1}(r_{S,v,j-1} + r_{v,T,j-1})/(q_{S,T,j-1} + q_{S,v,j-1}q_{v,T,j-1})$, i.e. if we went through (S, T) after, it corresponded to going from S to T directly before or from S to v to T and those options are not necessarily equally likely, so we need to multiply with the conditional probability of going through those edges. The second clause is saying that if we do state elimination and had (S, T) but not both (S, v) and (v, T) then the reward of (S, T) is as before. The third is saying that if we do state elimination and had both (S, v) and (v, T), but not (S, T) then the reward of (S, T) after is the reward of the path from S to v to T. The last is saying that if we did not do state elimination, then the reward remains unchanged.
The expression for $r_{v,S,j}$, for each S is

$$(x_{v,v,j} = 1 \wedge r_{v,S,j} = r_{v,S,j-1}) \vee x_{v,v,j} = 0$$

In words, if we did not do state elimination, then the length remains the same. If we did state elimination we do not care about $r_{v,S,j}$. We could as such equally have used the expression $r_{v,S,j} = r_{v,S,j-1}$, since we do not care about the value of $r_{v,S,j}$ if we did do state elimination, but like for the probability it seemed harder to follow.
The expression for $r_{S,v,j}$, for each $S \neq v$ is

$$(x_{v,v,j} = 1 \wedge r_{S,v,j} = r_{S,v,j-1}) \vee x_{v,v,j} = 0$$

In words, if we did not do state elimination, then the reward remains the same. If we did state elimination we do not care about $r_{S,v,j}$. Like above, we could do $r_{S,v,j} = r_{S,v,j-1}$ instead.

Finding the value. Once we did all the above iterations, we are left with a Markov chain where either state 1 is absorbing or it has no self-loop and has edges to absorbing states. All remaining states are absorbing. We want a variable v_s that, for each state, s is the value of the state (if it exists). For each state, $s \neq 1$, we have a variable v_s with expression

$$(x_{s,2N} = 1 \wedge v_s n_{s,s,2N-1} = r_{s,s,2N}) \vee x_{s,2N} = 0$$

In words, if the state exists, the mean-payoff value is $v_s = r_{s,s,2N}/n_{s,s,2N}$. Otherwise, if the state does not exist, we do not care.

For state 1, we have the variable v_1 with expression

$$(x_{1,1,2N} = 1 \wedge v_1 n_{1,1,2n} = r_{1,1,2N}) \vee (x_{1,1,2N} = 0 \wedge v_1 = \sum_{s \neq 1} x_{1,s,2N} q_{1,s,2N} v_s)$$

In words, the value of 1, if it is absorbing, is like the above. If it is not absorbing, it is for each other state s state 1 has an edge to, the probability of going to s from 1 times the value of s. Note that $x_{1,s,2N}$ is an indicator variable for whether the edge $(1, s)$ exists.

Proof of lemma. The correctness of the lemma follows from that each variable, both initially and based on the previous variables, is set correctly. There are $2N - 1$ iterations. In each iteration, we use 5 variables, 4 for edges and 1 for states. There can be at most N^2 edges and N states. Because we have variables for before and after each iteration (the before matches the ones for after the previous), we get a total of $8N^3 + 2N$ variables for this. Additionally, we use N variables for the values, getting us up to $8N^3 + 3N$. Each variable requires between 1 and 13 polynomials for below $100N^3 + 40N$ polynomials in total ($8 \cdot 13 = 104$, but many of the expressions are much smaller than the worst case). The maximum degree is $|A| \cdot b$ for each of these polynomials (for setting the initial variables). The worst case for the numbers in the coefficients comes from the input game of τ.

C PROOFS FOR SECTION 7

C.1 A proof of Claim 7.4

Claim 7.4. *Let $(v, \mathbf{m}, s) \to (v', \mathbf{m}', s')$ be an edge of the Markov chains M and M'.*

- *The difference in rewards in the edge $(v, \mathbf{m}, s) \to (v', \mathbf{m}', s')$ in M and M' is at most $C \cdot \delta(\sigma, \sigma')$.*
- *Let P and P' be the probability to go from $(v, \mathbf{m}, s)$ to $(v', \mathbf{m}', s')$ in M and M' respectively. Then, $\delta(P, P') \leq \delta(\sigma, \sigma')$.*

Proof. We first prove that the rewards differ additively. Without loss of generality, we assume that for every pair of game states $v, v' \in V$, actions vector $\mathbf{a} \in A^k$, and signal vector s, there exists a unique reward vector $\mathbf{r}$ such that $\Delta(v, \mathbf{a})(\mathbf{r}, s, v') > 0$. If the same action profile

produces different rewards on the same action, then we can replace all such transitions in the game by a single transition which produces the expected reward from all such possible transitions. Let

$$X \stackrel{\text{def}}{=} \sum_{a \in A^k \mid \Delta(v,a)(\mathbf{r},\mathbf{s}',v') > 0} \Delta(v,a)(\mathbf{r},\mathbf{s}',v')\mathbf{r}[i] \cdot \left(p(\sigma,v,\mathbf{m},s,\mathbf{m}',\mathbf{a}) - p(\sigma',v,\mathbf{m},s,\mathbf{m}',\mathbf{a})\right)$$

be the difference in rewards in M and M' for player i. For convenience, we write $p(a)$ instead of $p(\sigma,\mathbf{m},s,\mathbf{m}',\mathbf{a})$ and $p'(a)$ for $p(\sigma',\mathbf{m},s,\mathbf{m}',\mathbf{a})$.

We have that $\mathbf{r}[i] \in [-C,C]$ and we will next bound $p(a)-p'(a)$. We have that for all action profiles a, $p(a)-p'(a) = p(a)(1-p'(a)/p(a))$. There are two cases, either $p'(a) > p(a)$ or $p(a) > p'(a)$. If $p'(a) > p(a)$, then $1 - p'(a)/p(a) = -\delta(p(a),p'(a))$. Otherwise, if $p(a) > p'(a)$ then $\delta(p(a),p'(a)) = p(a)/p'(a) - 1$, implying that $p'(a)/p(a) = 1/(\delta(p(a),p'(a)) + 1)$ and thus that $1 - p'(a)/p(a) = \delta(p(a),p'(a))/(\delta(p(a),p'(a)) + 1)$. We have that $p(a)/p'(a) + 1) \leq \delta(p(a),p'(a))$ because $\delta(p(a),p'(a))$ is non-negative and we, therefore, divide with something bigger than 1. Hence, $p(a) - p'(a) \in [-p(a)\delta(p(a),p'(a)), p(a)\delta(p(a),p'(a))]$. Clearly, $\delta(p(a),p'(a)) \leq \delta(\sigma,\sigma')$, because it is a maximum over a set containing $\delta(p(a),p'(a))$. Thus, $|X| \leq \sum_{a=(a_1,\ldots,a_k)} C \cdot \Delta(v,a)(\mathbf{r},s,v') \cdot p(a)\delta(\sigma,\sigma') = C \cdot \delta(\sigma,\sigma') \sum_{a=(a_1,\ldots,a_k)} \Delta(v,a)(\mathbf{r},s,v')p(a)$, which is $\leq C \cdot \delta(\sigma,\sigma')$ as the sum of probabilities is ≤ 1.

We now prove that the transition probabilities differ multiplicatively. We have that $P = \sum_{a=(a_1,\ldots,a_k)} p(\sigma,\mathbf{m},s,\mathbf{m}',\mathbf{a}) \cdot \Delta(v,\mathbf{a})(r,\mathbf{s}',v')$ and similarly $P' = \sum_{a=(a_1,\ldots,a_k)} p(\sigma',\mathbf{m},s,\mathbf{m}',\mathbf{a}) \cdot \Delta(v,\mathbf{a})(r,\mathbf{s}',v')$. We will look at the sums term by term. We have three cases, either $P = P'$, $P > P'$ or $P < P'$. In the first case, we have that $\delta(P,P') = 0 \leq \delta(\sigma,\sigma')$. The two remaining cases we show just one as the other is analogous. Consider that $P > P'$ and thus $\delta(P,P') = P/P' - 1$. We have that

$$p(a)/p'(a) - 1 \leq \delta(\sigma,\sigma') \Rightarrow p(a) \leq (\delta(\sigma,\sigma') + 1)p'(a)$$

and therefore that

$$\begin{aligned}
\delta(P,P') + 1 = P/P' &= \frac{\sum_{a=(a_1,\ldots,a_k)} p(a) \cdot \Delta(v,a)(\mathbf{r},s,v')}{\sum_{a=(a_1,\ldots,a_k)} p'(a) \cdot \Delta(v,a)(\mathbf{r},s,v')} \\
&\leq \frac{\sum_{a=(a_1,\ldots,a_k)} p'(a)(\delta(\sigma,\sigma') + 1) \cdot \Delta(v,a)(\mathbf{r},s,v')}{\sum_{a=(a_1,\ldots,a_k)} p'(a) \cdot \Delta(v,a)(\mathbf{r},s,v')} \\
&= \delta(\sigma,\sigma') + 1
\end{aligned}$$

□

C.2 Approximating Markov Chains

We first recall some properties of u-bit floating point numbers $\mathbb{Q}(u)$ and the finite-precision variants of the addition, multiplication and division operations $\oplus^u, \oslash^u, \otimes^u$.

Proposition C.1.

(1) For every $x \in \mathbb{R}_{>0}$ and $u \in \mathbb{N}$ there exists $x' \in \mathbb{Q}(u)$ that is $(u,1)$-close to x.
(2) If x is (u,i)-close to y and y is (u,j)-close to z, then x is $(u,i+j)$-close to z.
(3) Let $x, \tilde{x}, y, \tilde{y}$ be non-negative numbers such that x is (u,i)-close to $\tilde{x}$ and y is (u,j)-close to $\tilde{y}$. Then, $x+y$ is $(u, max(i,j)+1)$-close to $\tilde{x}+\tilde{y}$, xy is $(u,i+j)$-close to $\tilde{x}\tilde{y}$, and x/y is $(u,i+j)$-close to $\tilde{x}/\tilde{y}$.
(4) If $x' \in \mathbb{Q}(u)$ is (u,j)-close to x and $y' \in \mathbb{Q}(u)$ is (u,j)-close to y, then $x' \oplus^u y'$ is $(u, \max(i,j)+1)$-close to $x+y$; $x' \oslash^u y'$ is $(u, ij+1)$-close to x/y; and $x' \otimes^u y'$ is $(u, i+j+1)$-close to xy;

We also recall some properties of probability distributions represented by approximately normalised representation.

Lemma C.2.

(1) For any probability distribution $q = (q_1, q_2, \ldots, q_m)$ there exist $(p_1, p_2, \ldots, p_m) \in \mathbb{P}(u)$ so that for all i, q_i and p_i are $(u, 2m+2)$-close.
(2) Suppose that $a_1, a_2, \ldots, a_m \in \mathbb{D}(u)$ and for all $i = 1 \ldots m$ let $p'_i = a_i \oslash^u \bigoplus^u_{j=1\ldots m} a_j$ and $p_i = p'_i / \sum_{j=1\ldots m} p'_j$. Then $(p'_1, p'_2, \ldots, p'_m)$ is an approximately normalised representation of $(p_1, p_2, \ldots, p_m) \in \mathbb{P}(u)$.

We recall the following Lemma of Solan [36]. This is very similar to our Lemma 7.3 in that it quantifies the change in value as a result of perturbations. Instead of perturbing strategies in MDPs, as we do in Lemma 7.3, this considers perturbations of transition probabilities *and reward functions* in Markov chains.

Lemma C.3 ([36], Theorem 4). *Given a Markov chain M and M' with the same states, let $dist(p_{ij}, p_{ij}) \leq \varepsilon$, $dist(r_{ij}, r'_{ij}) \leq C$, and $dist(d_{ij}, d'_{ij}) \leq C$. Then the mean payoff value of M and M' differ by at most $4\varepsilon NC$.*

We can now bound the error introduced by the imprecise loop elimination and state eliminations procedures.

Lemma 7.9 (Loop Elimination). *There is a polynomial δ_2 so that the following holds for all $u \in \mathbb{N}$.*
Let $M = (V, p, r, d)$ and $M' = (V, p', r', d')$ be Markov chains represented in u-bits so that M' results from M by eliminating a loop in state n as in Definition 5.3 but using u-bit floating point arithmetic. That is, $p'(n,n) = 0$; and for all $i, j \neq n$ let $p'(i,j) = p(i,j)$, $r'(i,j) = r(i,j)$,

$d'(i,j) = d(i,j)$ *and*

$$\begin{aligned} p'(n,j) &= p(n,j) \oslash (\oplus_{k\neq i} p(n,k)) \\ r'(n,j) &= ((r(n,j) \otimes p(n,j)) \oplus (p_{ii} \otimes r_{ii})) \\ &\quad \oslash (\oplus_{k\neq i} p(n,k)) \\ d'(n,j) &= ((d(n,j) \otimes p(n,j)) \oplus (p(n,n) \otimes d(n,n))) \\ &\quad \oslash (\oplus_{k\neq i} p(n,k)). \end{aligned}$$

Then,

1. *The smallest (negative) exponent among all floating point numbers in M' is at most one smaller than that in M.*
2. *Then $\left|\mathbb{E}_1^M[MPR] - \mathbb{E}_1^{M'}[MPR]\right| \leq \delta_2 2^{-u}$.*

Proof. Let $\hat{M}$ be the Markov chain obtain by the precise loop elimination algorithm. By Lemma 5.4, M has the same mean-payoff value as $\hat{M}$. Therefore, we bound the difference in mean-payoff value of M' and $\hat{M}$. Let N be the number of states in the Markov chain.

Since the probabilities in M and M' are given in approximately normalised representation, we write $p(i,j)$ and $p'(i,j)$ for the numbers giving the approximately normalised representation of the distributions $\tilde{p}(i,j)$ and $\tilde{p}'(i,j)$ respectively. We write $\hat{p}(i,j)$ to represent the probabilities in $\hat{M}$.

1. Each $p(i,j)$ is (u,N)-close to the actual probability distribution of M, which is $\tilde{p}(i,j)$ by Lemma C.2.
2. Each $\oplus_{k\neq i} p(n,k)$ is $(u, 2N-1)$-close to $\sum_{k\neq i} \tilde{p}(n,k)$ by $N-1$ applications of Item 4 of Proposition C.1 and item (1) above.
3. Each $p'(n,j)$ is $(u,(2N-1)N)$-close to $\hat{p}(n,j)$ by Item 4 of Proposition C.1 and items (1) and (2) above.
4. Each $\tilde{p}'(n,j)$ is (u,N)-close to $p'(n,j)$ by definition of approximately normalised representation and Lemma C.2.
5. Each $\tilde{p}'(n,j)$ is $(u, 2(N-1)N+N)$-close to $\hat{probm}(n,j)$ by Item 2 of Proposition C.1 and items (3) and (4), and therefore at least $(u, 2N^2)$-close.
6. Each $(r(n,j) \otimes p(n,j) \oplus (r(n,n)) \otimes p(n,n))$ is $(u, N+1)$-close to $r(n,j)\tilde{p(n,j)} + r(n,n)\tilde{p}(n,n)$ by item (1) above and Item 4 of Proposition C.1.
7. Each $r'(i,j)$ is $(u, 2(N-1)(N+1))$-close to $\hat{r}(i,j) = (r(n,j)\tilde{p(n,j)} + r(n,n)\tilde{p}(n,n))/\sum_{k\neq i}\tilde{p}(n,k)$ by Item 4 of Proposition C.1 applied to items (6) and (2), so at most $(u, 2N^2+N)$-close.
8. Each $d'(i,j)$ is $(u, 2N^2+N)$-close to $\hat{d}(i,j)$ by the same argument as item (7).

By Lemma C.3, we get that the mean-payoff value of M' differ from the value in $\hat{M}$ by at most $4\varepsilon NC$, where the ε is the maximum difference in the probabilities of $\hat{M}$ and M', and C is the maximum difference of rewards in $\hat{M}$ and M'. In our case, we have $\varepsilon = \left(\frac{1}{1-2^{-u-1}}\right)^{2(N)^2} - 1$. Since, $u \geq 1000N^2$, we have $\left(\frac{1}{1-2^{-u-1}}\right)^{2(N)^2} \leq 5(N^2)2^{-u}$, which gives $\varepsilon \leq 5(N^2)2^{-u}$. Similarly, we get $C \leq 5(N^2+N)2^{-u}$, which together gives $4N\varepsilon C \leq 100(N^4+N^3)2^{-u}$. Choosing $\delta_2 = 100(N^4+N^3)$ completes the proof of item (2).

The exponent for the probabilities are always negative as the values are ≤ 1. It is easy to see that in our updates $2\tilde{p}_{ij} \geq \tilde{p'}_{ij} \geq \tilde{p}_{ij}$. This implies that the negative exponents in the probabilities decreases by at most 1.

The rewards and durations can be assumed to be ≥ 1 and therefore the exponent can be assumed to be positive. Note that the rewards can be made > 1 by adding the smallest reward to each step and then subtracting it from the obtained value in the end. In a single step the rewards are less than twice the maximum reward in the input Markov chain. Therefore, the maximum exponent is at most 1 more than that in the input. A similar argument works for duration as well assuming that the durations in the input are all ≥ 1. □

Lemma 7.8 (State Elimination). *There is a polynomial δ_1 so that the following holds for all $u \in \mathbb{N}$.*

Let $M = (V, p, r, d)$ and $M' = (V, p', r', d')$ be Markov chains represented in u-bits so that M' results from M by eliminating a transient state n as in Definition 5.3 but using u-bit floating point arithmetic. That is, for all $i, j \neq n$,

$$\begin{aligned} p'(i,j) &= p(i,j) \oplus (p(j,n) \otimes p(n,j)) \\ r'(i,j) &= (p(i,j) \otimes r(i,j)) \\ &\quad \oplus ((p(i,n) \otimes p(n,j)) \otimes (r(i,n) \oplus r(n,j))) \\ d'(i,j) &= (p(i,j) \otimes d(i,j)) \\ &\quad \oplus ((p(i,n) \otimes p(n,j)) \otimes (d(i,n) \oplus d(n,j))) \end{aligned}$$

Then,

1. *The smallest (negative) exponent among all floating point numbers in M' is at most one smaller than that in M.*
2. *Then $\left|\mathbb{E}_1^M[MPR] - \mathbb{E}_1^{M'}[MPR]\right| \leq \delta_1 2^{-u}$.*

Proof. Let $\hat{M}$ be the Markov chain obtain by the precise state elimination algorithm. By Lemma 5.2, M has the same mean-payoff value as $\hat{M}$. Therefore, we bound the difference in mean-payoff value of M' and $\hat{M}$. We again estimate the relative distance between the precise and imprecise variables using the same notations as above. Let N be the number of states of the Markov chain.

(1) Each $p(i,j)$ is (u,N)-close to the actual probability distribution of M, which is $\tilde{p}(i,j)$ by Lemma C.2.
(2) Each $p'(i,j)$ is $(u,2N+2)$-close to $\hat{p}(i,j)$ by item (1) above and applying Item 4 of Proposition C.1.
(3) Each $r'(i,j)$ is $(u,2N+3)$-close to $\hat{r}(i,j)$ by applying Item 4 of Proposition C.1 to item (1). The dominating factor is the distance between $p(i,n) \otimes p(n,j) \otimes (r(i,n) \oplus r(n,j))$ and $\tilde{p(i,n)}p(\tilde{n,j})(r(i,n) \oplus r(n,j))$, which gives the $2N+3$ term.
(4) Similar to item (3), we have $d'(i,j)$ is $(u,2N+3)$-close to $\hat{d}(i,j)$.

Instead of using the bound $2N+3$ obtained above, we instead use $3N$ for the sake of brevity, as $(u,2N+3)$-closeness implies $(u,3N)$-close assuming $N \geq 3$. By Lemma C.3, we get that the mean-payoff value of M' differ from the value in $\hat{M}$ by at most $4N\varepsilon C$, where the ε, $\varepsilon = \left(\frac{1}{1-2^{-u-1}}\right)^{3N} - 1$. Since $u \geq 1000N^2$, we have $\left(\frac{1}{1-2^{-u-1}}\right)^{3N} \leq 1 + 7N2^{-u}$ which gives $\varepsilon \leq 1 + 7N2^{-u}$. We get the same bound on C and therefore, $4N\varepsilon C \leq 200N2^{-u}$. Choosing $\delta_1 = 200N$ completes the proof of item (2).

The increase in positive and decrease in negative exponents can be shown to be at most 1 by the same argument as the loop-elimination case. □