Exact Reduction of Huge Action Spaces in General Reinforcement Learning

Sultan J. Majeed[†] sultan.pk
Research School of Computer Science, ANU
Marcus Hutter hutter1.net
Google DeepMind & Research School of Computer Science, ANU

Abstract

The reinforcement learning (RL) framework formalizes the notion of learning with interactions. Many real-world problems have large state-spaces and/or action-spaces such as in Go, StarCraft, protein folding, and robotics or are non-Markovian, which cause significant challenges to RL algorithms. In this work we address the large action-space problem by sequentializing actions, which can reduce the action-space size significantly, even down to two actions at the expense of an increased planning horizon. We provide explicit and exact constructions and equivalence proofs for all quantities of interest for arbitrary history-based processes. In the case of MDPs, this could help RL algorithms that bootstrap. In this work we show how action-binarization in the non-MDP case can significantly improve Extreme State Aggregation (ESA) bounds. ESA allows casting any (non-MDP, non-ergodic, history-based) RL problem into a fixed-sized non-Markovian state-space with the help of a surrogate Markovian process. On the upside, ESA enjoys similar optimality guarantees as Markovian models do. But a downside is that the size of the aggregated state-space becomes exponential in the size of the action-space. In this work, we patch this issue by binarizing the action-space. We provide an upper bound on the number of states of this binarized ESA that is logarithmic in the original action-space size, a double-exponential improvement.

Contents

1 Introduction

The reinforcement learning (RL) setting can be described by an agent-environment interaction [**?**]. The agent Π has an action-space $\mathscr{A}$ to choose its actions from while the environment P reacts to the action by dispensing an observation and a reward from the sets $\mathscr{O}$ and $\mathscr{R} \subseteq \mathbb{R}$,

[†]The contact author.

respectively, see Figure 1. For simplicity, we assume that these sets are finite and hence the rewards are bounded. Even with these restrictions, the problem of RL does not trivialize, i.e. the agent can not learn the optimal behavior without further structure. Under a suitable definition of the "state" of environment, the resultant set of states might be huge or even infinite [?].

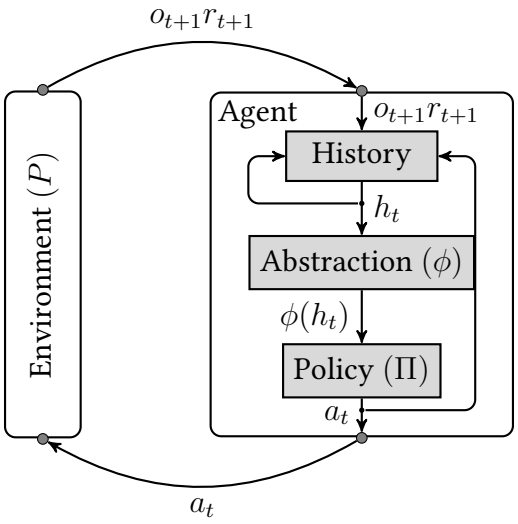

Figure 1: The agent-environment interaction.

The problem of RL is plagued with the curse of dimensionality. The sizes of an appropriately defined set of states[1] and action-space play an important role in the choice of algorithms, architectures and solution techniques used to solve the task [?].

It is usually required to produce a relatively small set of states to make the problem tractable [?]. There are many ways to achieve such approximations/abstractions, e.g. state aggregation [? ? ?], homomorphism [?], linear function approximation [?], or neural networks [?] just to name a few. The usual assumption for such abstractions is that they try to produce a Markovian representation of the environment, which is known as a Markov Decision Process (MDP) [?]. In an MDP the most recent (abstract) observation is sufficient to predict *any* future event[2], but not all events are equally valuable. Some events might not lead to a high rewarding state and/or some distinctions are not really necessary to perform well [?]. For example, the agent might end up experiencing two completely different streams of observations with the same reward structure. An algorithm which tries to produce a Markovian representation would try to make this "unnecessary" distinction.

A lot can be achieved in terms of the representation power if an algorithm only makes "useful" distinctions, i.e. the distinctions (or states) which respect the reward structure [? ? ?]. In some cases, such "useful" but non-Markovian abstractions reduce the effective state-space dramatically. Usually, a smaller state-space facilitates faster learning [? ?].

The usual methods of state or action-space reductions either (1) reduce the problem to a fixed size where the quality of reduction deteriorates as the original problem becomes more complicated, or (2) provide a problem-specific reduction which usually grows, albeit much slower, as the original problem grows [?]. In Markovian abstractions the size of the state-space grows with the size of the observation and reward spaces. For example, if an MDP

[1] We prefer not to call the set of observations $\mathcal{O}$ the set of states. This set becomes a set of states under strong assumptions, see Section 5 for more details on this distinction.

[2] An event is any set of action-observation-reward sequences.

abstraction produces states from some low-resolution images then we need more states to handle high-resolution versions of the input images because the high-resolution images need a bigger transition matrix to predict the next image. However, it is perfectly plausible that the increased resolution might not be "useful" to achieve better rewards.

To the best of our knowledge, extreme state aggregation (ESA), a non-MDP abstraction framework, is the only method which provides a provable upper bound on the size of required state-space uniformly[3] for all problems [**?**]. However, a *downside* of ESA is that the size of the aggregated state-space is *exponential* in the size of the action-space, see Theorem 5.2. In this paper, we move the research further in this direction. We provide a variant of ESA that can help provide much more compact representations as compared to MDP abstractions. Our approach improves the key upper bound on the size of the state-space in the original ESA framework.

The key trick to achieve this improvement is to sequentialize the actions. Often $\mathscr{A}$ already has a natural vector structure $\mathscr{B}^d$, e.g. real valued activators in robotics ($\mathscr{B} = \mathbb{R}$) or (padded) words ($\mathscr{B} = \{a, \ldots, z, \textvisiblespace\}$), or more generally $\mathscr{B}_1 \times \ldots \times \mathscr{B}_d$, where $\mathscr{B}$ denotes a finite set of *decision symbols*. In this case, sequentialization is natural, but one may further want to binarize $\mathscr{B}$ to $\mathbb{B}^{d'}$ esp. for ESA (Theorem 5.3). If actions are (converted to) $\mathscr{B}$-ary strings, the RL agent could execute the action "bits" sequentially with fictitious dummy observations in-between.

The example in Figure 2 provides a naive way of *sequentializing* the actions in an MDP. Apparently, it might seem that such sequentialization of the action-space would be of no help, as the state-space would blow up, and it is simply shifting the problem from the actions to the states. However, we prove that this can be avoided. Most importantly, the universal upper bound on the effective state-space of ESA remains valid. Our scheme of sequentializing the actions achieves a *double exponentially* improved bound; compare Theorem 5.3 with Theorem 5.2.

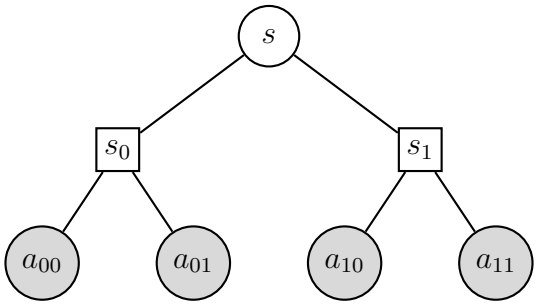

Figure 2: A simple sequentialization example in an MDP. To see how the actions sequentialized, consider an agent which has to choose among four alternatives, e.g. $\mathscr{A} = \{a_{00}, a_{01}, a_{10}, a_{11}\}$. Let the agent receive a state signal s from the environment. It first decides between a partition of actions, say two actions each, $\{a_{00}, a_{01}\}$ and $\{a_{10}, a_{11}\}$. *After* it has decided on the bifurcation, the *extended* state becomes s_x, where x is the decision of the first stage. Now the agent *on* this *extended* state s_x makes its second decision to choose from the *short-listed* set of actions. This way, the agent only selects among *two* alternatives at each stage by *tripling* the effective state-space.

Along the way, we also establish some other key results, which are interesting and useful on their own. We provide explicit and exact constructions and equivalence proofs for all quantities of interest (Section 4) for arbitrary history-based processes, which are then used

[3]Which depends only on the size of the action-space, discount factor and the optimality gap.

to double-exponentially improve the previous ESA bound (Theorem 5.3). In the special case of MDPs, we show that through a sequentialized scheme (of augmenting observations with partial decision vectors) the resultant "sequentialized process" preserves the Markov property (Theorem 4.2), which should help RL algorithms that bootstrap, though demonstrating or proving this is left for future work. Moreover in Theorem 4.6, we prove that the stipulated sequentialization scheme preserves near-optimality, i.e. a near-optimal policy of the sequentialized process is also near-optimal in the original process.

The rest of the paper is organized as follows. Section 2 puts down the necessary notation. In Section 3, we formally set up the problem. A framework to sequentialize actions is provided in Section 4, along with other key results (Theorems 4.2 and 4.6). In Section 5 we combine our sequentialization framework with ESA to improve the upper bound on the size of the state-space (Theorem 5.3). We conclude the paper in Section 6 with an outlook.

2 Notation

This paper is notation heavy, but we use a consistent notation through out. The set of natural numbers is $\mathbb{N} := \{1, 2, \dots\}$, $\mathbb{B} := \{0, 1\}$ is a set of binary symbols, and $\mathbb{R}$ is the set of reals. We denote by $\triangle(X)$ the set of probability distributions over any set X. The concatenation of two objects (or strings) is expressed through juxtaposition, e.g. xy is a concatenation of x and y. We express a finite string with boldface, e.g. $\boldsymbol{x} = x_1 x_2 \dots x_{|\boldsymbol{x}|}$ where $|\cdot|$ is used to denote the length or cardinality of the object. The individual members of a string or a vector may be accessed as $\boldsymbol{x}_i = x_i$ for any $i \leq |\boldsymbol{x}|$. A substring of length $i \leq |\boldsymbol{x}|$ is denoted as $\boldsymbol{x}_{\leq i} = x_1 x_2 \dots x_i$ and $\boldsymbol{x}_{<i} = x_1 x_2 \dots x_{i-1}$. We interchangeably use the same notation for vectors and strings, e.g. $\boldsymbol{x} \in \mathscr{B}^d$ is a d-dimensional $\mathscr{B}$-ary decision vector which may also be expressed as a string. This choice simplifies the notation and saves redundant variables. If a variable is time-indexed, we express the continuation of the variable with a prime on it, e.g. if $x := x_t$ then $x' := x_{t+1}$ where $:=$ denotes equality by definition. A small scaler value (usually the error tolerance) is denoted by $\varepsilon > 0$. A different member of the same set is expressed with a dot on it, e.g. $x, \dot{x} \in \mathscr{B}$. We express the fact of $\boldsymbol{x}$ being a prefix of $\boldsymbol{y}$ by $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ or $\boldsymbol{y} \sqsupseteq \boldsymbol{x}$. Moreover, $\overline{\boldsymbol{xy}}$ represents a vector that is point-wise joined, i.e. $\overline{\boldsymbol{xy}}_i := \boldsymbol{x}_i \boldsymbol{y}_i$.

3 Problem Setup

This section provides the formal foundations for us to build up to the main result of this work. We formulate the problem of RL from the ground up without starting from the usual Markovian assumption [**?**]. We formalize a history-based RL setup. After formalizing the (general) RL problem, we set up the scheme of sequentializing the decision-making process to reduce the effective action-space for the agent. Especially for our main result about ESA, we sequentialize the action-space to *binary* decisions.

Although this work assumes very little about the RL problem, we assume that the size of the action-space is finite and $|\mathscr{A}| = |\mathscr{B}|^d$ for some $d \in \mathbb{N}$. The latter assumption is not restrictive, as we can extend the set of actions by repeating some of the actions. It is important to note that these repeated actions should be labeled distinctly. This way we can have a bijection between the original (extended) action-space and the sequentialized one. For example, let an action set be $\{a_1, a_2, a_3, a_4, a_5\}$. One possible extended set, with repetition, for $|\mathscr{B}| = 2$ and $d = 3$ is $\mathscr{A} := \{a_1, a_2, a_3, a_4, a_5, a_{5_1}, a_{5_2}, a_{5_3}\}$. Where, the actions a_{5_i} for $i \leq 3$ are functionally the same as a_5, i.e. taking a_5 or any a_{5_i} action has the same effect, but they are labeled

distinctly.

Note that continuous action-spaces could be approximately sequentialized/binarized by using the binary expansion of reals to some desired precision, say δ. Our main bound will only depend logarithmically on δ.

3.1 General Reinforcement Learning

We consider a general reinforcement learning (GRL) setup where the agent keeps the complete history of interaction [**?**]. The (infinite[4]) interaction produces an infinite history. Recall that $\mathscr{O}, \mathscr{R}$ and $\mathscr{A}$ represent some *finite* sets of observations, rewards and actions, respectively. This also implies that the rewards are bounded. The set of all finite histories[5] is denoted by

$$\mathscr{H} := \bigcup_{t=1}^{\infty} \underbrace{\mathscr{O} \times \mathscr{R} \times \mathscr{A} \times \ldots \times \mathscr{O} \times \mathscr{R} \times \mathscr{A}}_{(t-1)-\text{step interactions}} \times \mathscr{O} \times \mathscr{R} \tag{1}$$

which is used to express most of the quantities in our setup, e.g. the environment, agent, and value functions. Note that the history set $\mathscr{H}$ does not contain the empty history. This is a design choice we make to be consistent with the standard RL setup [**?**] where the initial state (in our case the initial observation and reward) is chosen by some "initial" distribution.

Formally, the environment P is a (conditional) probability function such that $P : \mathscr{H} \times \mathscr{A} \to \triangle(\mathscr{O} \times \mathscr{R})$. Similarly, the agent is also expressible as a (conditional) distribution on the action-space, i.e. $\Pi : \mathscr{H} \to \triangle(\mathscr{A})$. For a fixed policy Π the expected discounted future sum of rewards is the value of the policy. At any history $h \in \mathscr{H}$ and action $a \in \mathscr{A}$ the action-value function (or Q-function) is expressed as

$$Q^{\Pi}(h, a) := \sum_{o'r'} P(o'r'|ha) \left(r' + \gamma V^{\Pi}(hao'r')\right) \tag{2}$$

where $V^{\Pi}(h) := \sum_a Q^{\Pi}(h, a)\Pi(a|h)$ is the value function of Π and $0 \leq \gamma < 1$ is the discount factor. Equation (2) is known as the Bellman equation (BE). The optimal behavior (or policy) is the one which achieves the maximum value for all histories, i.e. $\Pi^*(h) :\in \arg\max_{\Pi} V^{\Pi}(h)$. The optimal value (and action-value) functions, $V^* := V^{\Pi^*}$ and $Q^* := Q^{\Pi^*}$, of this optimal policy satisfy the following optimal Bellman equation (OBE) [**? ?**].

$$Q^*(h, a) := \sum_{o'r'} P(o'r'|ha) \left(r' + \gamma V^*(hao'r')\right) \tag{3}$$

where $V^*(h) := \max_a Q^*(h, a)$. The agent defined in this sub-section works with the original action-space $\mathscr{A}$ and keeps the histories from $\mathscr{H}$. In the next sub-section, we formulate another agent which only works in the "sequentialized" action-space, i.e. it takes decisions in a sequence of $\mathscr{B}$-ary choices, and responds *only* to the histories generated by this $\mathscr{B}$-ary interaction, see Figure 3. In the extreme case, this agent may only take binary decisions by sequentializing the action-space to binary sequences, i.e. $\mathscr{B} = \mathbb{B}$.

[4]For simplicity, we assume the interaction never stops. We do not consider the case where the agent or the environment can stop responding. It complicates the modeling beyond the scope of this work.

[5]Note that this set (of underlying "state" space) is (countably) infinite.

3.2 Sequential Decisions

We want to transform the action-space into a sequence of $\mathscr{B}$-ary decision code words, which are decided sequentially. To map the actions between the original action-space and the sequentialized decision-space, we define a pair of encoder and decoder functions. Let C be any encoding function that maps the actions to a $\mathscr{B}$-ary decision code of length d, i.e. $C : \mathscr{A} \to \mathscr{B}^d$. A decoder function $D : \mathscr{B}^d \to \mathscr{A}$ sends the $\mathscr{B}$-ary decision sequences generated by C back to the actions in the (original) action-space. In this work, the choices of C and D do not matter[6] as long as they are bijections such that $D(C(\mathscr{A})) = \mathscr{A}$.

This sequentialization of the action-space changes the interaction history. The generated histories are no longer members of $\mathscr{H}$. The goal of this paper is to argue that an agent can still work with the sequentialized histories only. The agent can plan, learn and interact with the environment using $\mathscr{B}$-ary actions and keeping sequentialized histories. Hence, the agent can be agnostic to the original action-space and with the state provided through an appropriate abstraction it can only take $\mathscr{B}$-ary decisions at every time step, see Figure 3.

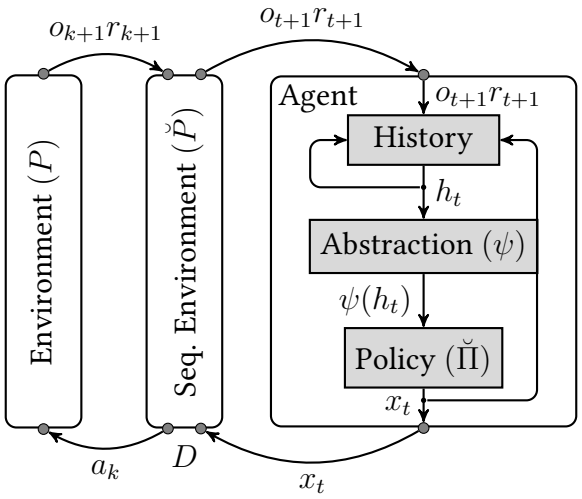

Figure 3: The agent-environment interaction through the sequentialization scheme. Note that the sequentialized-environment block (or a $\mathscr{B}$-ary "mock") manages two different time-scales t and k. It is simply a buffer block which knows (de)coders C and D (see text for details). It buffers the input $\mathscr{B}$-ary actions and dispatches the buffered observation and reward. Once a complete $\mathscr{B}$-ary decision sequence is produced by the agent the $\mathscr{B}$-ary mock decodes the encoded actions to the original environment to continue the interaction loop. We can consider this sequentialized environment as a "middle layer" between the agent and the original environment.

We construct a history transformation function which maps the original histories from $\mathscr{H}$ to some sequentialized histories in $\breve{\mathscr{H}}$, where

$$\breve{\mathscr{H}} := \bigcup_{t=1}^{\infty} \underbrace{\mathscr{O} \times \mathscr{R} \times \mathscr{B} \times \ldots \times \mathscr{O} \times \mathscr{R} \times \mathscr{B}}_{(t-1)-\text{step interactions}} \times \mathscr{O} \times \mathscr{R} \tag{4}$$

It is worth noting that $\breve{\mathscr{H}}$ does not (directly) contain any information about $\mathscr{A}$, cf. Equation (1). The agent experiencing histories from this set would not be aware of $\mathscr{A}$.

[6]The choice could matter in practical implementation of such agents. For example, a clever choice of such functions might produce sparse $\mathscr{B}$-ary decision sequences for the optimal actions, hence it may facilitate in learning such optimal $\mathscr{B}$-ary decision sequences.

Definition 3.1 (History transformation function)**.** *A history transformation function is expressed with* $g : \mathscr{H} \to \breve{\mathscr{H}}$*. The map is recursively defined for any history* h*, action* a*, next observation* o' *and next reward* r' *as*

$$g(hao'r') := g(h)\boldsymbol{x}_1 or_\perp \boldsymbol{x}_2 or_\perp \dots \boldsymbol{x}_d o'r' \text{ and } g(e) := e \tag{5}$$

where $\boldsymbol{x} := C(a)$*,* o *is the last observation of the history* h*,* e *denotes the "initial" history*[7]*, and* $r_\perp$ *is any fixed real-value. In this work, we assume*[8] *that* $r_\perp \in \mathscr{R}$ *and* $r_\perp = 0$*.*

In the above construction, we chose to repeat the last observation o in between the real interactions with the environment. This is not the only possible choice, we can choose a *dummy* observation $o_\perp \in \mathscr{O}$ instead without affecting the claims. For brevity, we define $\boldsymbol{o}$ and $\boldsymbol{r}_\perp$ as d-dimensional constant vectors of o and $r_\perp$, respectively. These vectors are then "welded" with $\boldsymbol{x}$ to succinctly replace $x_1 or_\perp \dots x_i or_\perp$ with $\overline{\boldsymbol{x}\boldsymbol{o}\boldsymbol{r}_\perp}_{\leq i}$. Note that we do not sequentialize the observations. It can be done, but we believe it is not useful in any way.

However, if the original process P is an MDP, i.e. the most recent observation is the state of P, then there is another interesting option possible for $o_\perp$: extend the observation space $\mathscr{O}$ with $\mathscr{O} \times \cup_{i=0}^{d-1} \mathscr{B}^i =: \widetilde{\mathscr{O}}$, and let the $\mathscr{B}$-ary mock dispatch an appropriate observation at every partial $\mathscr{B}$-ary decision vector $\boldsymbol{x}_{<i}$ as:

$$\widetilde{o}_\perp := (o, \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{i-1}) \in \widetilde{\mathscr{O}} \tag{6}$$

It is not hard to show that with this sequentialization scheme the resultant sequentialized decision process is also an MDP over $\widetilde{\mathscr{O}}$, see Theorem 4.2. By doing so, we end up with a state-space of size $|\widetilde{\mathscr{O}}| = |\mathscr{O}|(|\mathscr{A}| - 1) \leq |\mathscr{O} \times \mathscr{A}|$. It is clear that this recasting of the original problem might not be very helpful for some Monte-Carlo like tree search methods, however, it might significantly improve the performance of some temporal-difference like algorithms,, e.g. Q-learning [**?**], when applied to huge action-spaces.

Note that g is injective, but it may not be a bijection. There are many sequentialized histories $\tau \in \breve{\mathscr{H}}$ which are not mapped by g, i.e. there does not exist any history in $\mathscr{H}$ such that $\tau = g(h)$. For such sequentialized histories we define $g^{-1}(\tau) := \perp$, which formally allows us to talk about g^{-1} without worrying about it being undefined on some arguments. The choice of this definition is not important. As a matter of fact, there is no particular significance of the symbol $\perp$. What makes this choice insignificant is the fact that the environment does not react until the agent has taken d $\mathscr{B}$-ary actions. Some histories not covered by g are such "partial" sequentialized histories where the actual environment does not react. Note that the rewards of the sequentialized setup are zero ($r_\perp := 0$) unless the sequentialized history length is a multiple of d, i.e. a "complete" sequentialized history. See Figure 4 for an example sequentialized/binarized setup for $\mathscr{B} = \mathbb{B}$ and $d = 2$.

Any agent which interacts with the environment through this sequentialized scheme would effectively experience the following sequentialized environment.

Definition 3.2 (Sequentialized environment)**.** *For any* $\mathscr{B}$*-ary action* $x \in \mathscr{B}$*, sequentialized history* $\tau \in \breve{\mathscr{H}}$*, and any partial extension* $\overline{\boldsymbol{x}\boldsymbol{o}\boldsymbol{r}_\perp}_{<i}$ *for* $i \leq d$ *the probability of receiving* o' *and*

[7] The initial history $e \in \mathscr{O} \times \mathscr{R}$ is similar to the initial state in standard RL. It is dispatched by environment without any input at the start.

[8] This assumption is not much of a restriction, if $r_\perp \notin \mathscr{R}$ then we can extend the reward space by $\mathscr{R} \cup \{r_\perp\}$.

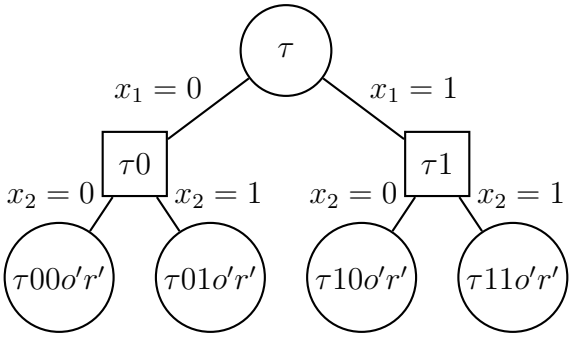

Figure 4: A simple sequentialization/binarization example in a deterministic history-based process. The $\mathscr{B}$-ary/binary decisions are on the edges. For brevity, we do not represent $o_\perp$ and $r_\perp$ in the figure. For example, it should be apparent that $\tau 1 o_\perp r_\perp \equiv \tau 1$. The circles represent complete histories while the squares indicate partial histories.

r' as the next observation and reward is as follows:

$$\breve{P}(o'r'|\tau\overline{\boldsymbol{xor}_\perp}_{<i}x) := \begin{cases} P(o'r'|ha) & \text{if } \tau\overline{\boldsymbol{xor}_\perp}_{<i}xo'r' = g(hao'r') \\ 1 & \text{if } o'r' = or_\perp \\ & \text{and } g^{-1}(\tau\overline{\boldsymbol{xor}_\perp}_{<i}xo'r') = \perp \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where P is the actual environment.

As highlighted before, the history h can not be empty, so the above definition is well-defined.

The next step is to define the (action) value functions for this sequentialized agent-environment interaction. Let $\breve{\Pi}$ be a policy such that $\breve{\Pi} : \breve{\mathscr{H}} \to \triangle(\mathscr{B})$. Then, we define the (action) value functions similar to the original agent-environment interaction case. For any $\tau \in \breve{\mathscr{H}}$ and $x \in \mathscr{B}$, the action-value function is defined as

$$\breve{Q}^{\breve{\Pi}}(\tau, x) := \sum_{o'r'} \breve{P}(o'r'|\tau x) \left(r' + \lambda \breve{V}^{\breve{\Pi}}(\tau x o'r') \right) \tag{8}$$

where $\breve{V}^{\breve{\Pi}}(\tau) := \sum_{x \in \mathscr{B}} \breve{Q}^{\breve{\Pi}}(\tau, x)\breve{\Pi}(x|\tau)$ and λ is the discount factor of this sequentialized problem. Similar to the original optimal (action) value functions, $\breve{Q}^*$ and $\breve{V}^*$ denote the optimal (action) value functions of the sequentialized problem. The discount factor λ plays an important role in trading off the size of the action-space with the planning horizon. Recall that the size of the original action-space is $|\mathscr{A}| = |\mathscr{B}|^d$. Therefore, if the agent has to make d $\mathscr{B}$-ary decisions for each original action the discount factor after d $\mathscr{B}$-ary actions should be γ, i.e. $\lambda^d = \gamma$. This implies that $\lambda = \gamma^{1/d} < 1$ as $\gamma < 1$ and $d < \infty$.

This completes the problem setup. We have defined an agent $\breve{\Pi}$ which only makes $\mathscr{B}$-ary decisions and reacts to sequentialized histories, see Figure 3. As expected, the set of sequentialized histories $\breve{\mathscr{H}}$ is blown out in comparison with $\mathscr{H}$. However, in Section 5, we show that, under certain non-Markovian abstractions of either $\mathscr{H}$ or $\breve{\mathscr{H}}$, this expansion is not "harmful".

4 Sequentialized Processes and Values

In this section we formally define the sequentialized process and related value functions. But first we need a couple of important quantities to state our main results. For any $\mathscr{B}$-ary vector $\boldsymbol{x} \in \mathscr{B}^i$ where $i \leq d$, we define $\mathscr{A}(\boldsymbol{x}) := \{a \in \mathscr{A} : \boldsymbol{x} \sqsubseteq C(a)\}$ a *restricted* set of actions. Moreover, for any history h, an action-value function maximizer on this restricted set is defined as $\Pi^*(h, \boldsymbol{x}) \in \arg\max_{a \in \mathscr{A}(\boldsymbol{x})} Q^*(h, a)$.

We start off the section by noting an important relationship between the sequentialized process and the original process.

Proposition 4.1 (Sequentialized Process)**.** *For any $o' \in \mathscr{O}, r' \in \mathscr{R}, h \in \mathscr{H}$ and $D(\boldsymbol{x}) =: a \in \mathscr{A}$, the following relationship holds between $\breve{P}$ and P:*

$$\breve{P}(o'r'|g(h)\overline{\boldsymbol{xor}}_{\perp<d}\boldsymbol{x}_d) = P(o'r'|ha) \tag{9}$$

Proof. The proof trivially follows from Definition 3.2 by evaluating the definition at $i = d$ with $D(\boldsymbol{x}_{<d}\boldsymbol{x}_d) = a$. □

When the original process is an MDP then there exists a sequentialization scheme such that the sequentialized process is also Markovian.

Theorem 4.2 (Sequentialization preserves Markov property)**.** *If P is an MDP over $\mathscr{O}$, and the observations from the $\mathscr{B}$-ary mock are $\widetilde{\mathscr{O}} := \mathscr{O} \times \cup_{i=0}^{d-1} \mathscr{B}^i$, then $\breve{P}$ is also an MDP over $\widetilde{\mathscr{O}}$.*

Proof. In the case of augmenting the observation space, the definition of $\breve{P}$ becomes slightly more verbose than Definition 3.2 as $o_\perp$ is different for each partial history as defined in Equation (6).

$$\breve{P}(\widetilde{o}'r'|\tau\overline{\boldsymbol{x}\widetilde{\boldsymbol{o}}\boldsymbol{r}}_{\perp<i}x) := \begin{cases} P(o'r'|ha) & \text{if } \tau\overline{\boldsymbol{x}\widetilde{\boldsymbol{o}}\boldsymbol{r}}_{\perp<i}x\widetilde{o}'r' = g(hao'r') \\ 1 & \text{if } \widetilde{o}'r' = o\boldsymbol{x}_{<i}xr_\perp \\ & \text{ and } g^{-1}(\tau\overline{\boldsymbol{x}\widetilde{\boldsymbol{o}}\boldsymbol{r}}_{\perp<i}x\widetilde{o}'r') = \perp \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

for any $i \leq d, \widetilde{o}, \widetilde{o}' \in \widetilde{\mathscr{O}}$, and $o \in \mathscr{O}$ is the most recent observation in h. At any h the sufficient information is o, so $P(o'r'|ha) \equiv P(o'r'|oa)$. Therefore, from the above (expanded) definition of $\breve{P}$, it is clear that:

$$\breve{P}(\widetilde{o}'r'|\tau\overline{\boldsymbol{x}\widetilde{\boldsymbol{o}}\boldsymbol{r}}_{\perp<i}x) \equiv \breve{P}(\widetilde{o}'r'|o\boldsymbol{x}_{<i}x) = \breve{P}(\widetilde{o}'r'|\widetilde{o}x)$$

hence proves the proposition. □

The following proposition proves that the action-values of the "partial" histories of the sequentialized problem are related. This fact later helps us to show that these action-value functions respect the Q-uniform structure of the original environment.

Proposition 4.3 ($\breve{Q}^*$ max-relationship)**.** *For any sequentialized history $\tau \in \breve{\mathscr{H}}$ such that $g^{-1}(\tau) \in \mathscr{H}$, the following holds*

$$\max_{x \in \mathscr{B}} \breve{Q}^*(\tau, x) = \lambda^{d-1} \max_{\boldsymbol{x} \in \mathscr{B}^d} \breve{Q}^*(\tau\overline{\boldsymbol{xor}}_{\perp<d}, \boldsymbol{x}_d) \tag{11}$$

Proof. The proof is straight forward. We successively apply the definition of $\breve{Q}^*$.

$$\begin{aligned}
\max_{x_1 \in \mathscr{B}} \breve{Q}^*(\tau, x_1) &= \max_{x_1 \in \mathscr{B}} \sum_{o'r'} \breve{P}(o'r'|\tau x_1) \\
&\qquad \left(r' + \lambda \max_{x_2 \in \mathscr{B}} \breve{Q}^*(\tau x_1 o' r', x_2) \right) \\
&\overset{(a)}{=} \lambda \max_{x_1 \in \mathscr{B}} \max_{x_2 \in \mathscr{B}} \breve{Q}^*(\tau x_1 o r_\perp, x_2) \\
&\quad \vdots \text{ (continue unrolling for } d-1\text{-steps)} \\
&= \lambda^{d-1} \max_{\boldsymbol{x} \in \mathscr{B}^d} \breve{Q}^*(\tau \overline{\boldsymbol{xor}}_{\perp<d}, \boldsymbol{x}_d) \qquad (12)
\end{aligned}$$

where (a) follows from the definition of $\breve{P}$ and the fact that $r' = r_\perp = 0$ when $\breve{P} \neq 0$. □

Now, using Proposition 4.3 we can prove a relationship between the action-value functions of the actual environment and the sequentialized environment.

Lemma 4.4 ($\breve{Q}^*$ $\boldsymbol{x}$-relationship)**.** *For any history h with the corresponding sequentialized history $\tau = g(h)$ and $\mathscr{B}$-ary decision vector $\boldsymbol{x} \in \mathscr{B}^d$, the following holds for any $i \leq d$.*

$$\breve{Q}^*(\tau \overline{\boldsymbol{xor}}_{\perp<i}, \boldsymbol{x}_i) = \gamma^{\frac{d-i}{d}} Q^*(h, \Pi^*(h, \boldsymbol{x}_{\leq i}))$$

Proof. Before we prove the general result, we show that the result holds for $i = d$, i.e. the sequentialized problem has same optimal action-values at the "real" decision steps. Note that $\Pi^*(h, \boldsymbol{x}_{\leq d}) = D(\boldsymbol{x})$. Let $\boldsymbol{x} := C(a)$ and $\tau := g(h)$. Using the fact that $r_\perp = constant = 0$, we get

$$\begin{aligned}
f_{r_\perp}(h, a) &:= \breve{Q}^*(\tau \overline{\boldsymbol{xor}}_{\perp<d}, \boldsymbol{x}_d) \\
&\overset{(a)}{=} \sum_{o'r'} \breve{P}(o'r'|\tau \overline{\boldsymbol{xor}}_{\perp<d} \boldsymbol{x}_d) \left(r' + \lambda \max_{x'} \breve{Q}^*(\tau \overline{\boldsymbol{xor}}_{\perp<d} \boldsymbol{x}_d o' r', x') \right) \\
&\overset{(b)}{=} \sum_{o'r'} P(o'r'|ha) \left(r' + \lambda \max_{x'} \breve{Q}^*(\tau \overline{\boldsymbol{xor}}_{\perp<d} \boldsymbol{x}_d o' r', x') \right) \\
&\overset{(c)}{=} \sum_{o'r'} P(o'r'|ha) \left(r' + \lambda^d \max_{\boldsymbol{x}' \in \mathscr{B}^d} \breve{Q}^*(\tau \overline{\boldsymbol{xor}}_{\perp<d} \boldsymbol{x}_d o' r' \overline{\boldsymbol{xor}}'_{\perp<d}, \boldsymbol{x}'_d) \right) \\
&\overset{(d)}{=} \sum_{o'r'} P(o'r'|ha) \left(r' + \gamma \max_{a' \in \mathscr{A}} f_{r_\perp}(hao'r', a') \right) \qquad (13)
\end{aligned}$$

where (a) is just Equation (8) with the optimal policy, (b) follows by Proposition 4.1, (c) is given by Proposition 4.3, (d) is true by rearranging the argument, the definition of $f_{r_\perp}$ and by using the relation $\lambda^d = \gamma$. Note that Equation (13) is the OBE of the original problem, see Equation (3). The solution of the OBE is unique [**?**], hence $f_{r_\perp}$ is indeed Q^*.

Having proved the claim for $i = d$, we show that it also holds for any $i < d$.

$$\begin{aligned}
\breve{Q}^*(\tau\overline{\boldsymbol{xor}}_{\perp<i}, \boldsymbol{x}_i) &\overset{(a)}{=} \sum_{o'r'} \breve{P}(o'r'|\tau\overline{\boldsymbol{xor}}_{\perp<i}\boldsymbol{x}_i)\left(r' + \lambda \max_{x_{i+1}} \breve{Q}^*(\tau\overline{\boldsymbol{xor}}_{\perp<i}\boldsymbol{x}_i o'r', x_{i+1})\right) \\
&\overset{(b)}{=} \lambda \max_{x_{i+1}} \breve{Q}^*(\tau\overline{\boldsymbol{xor}}_{\perp<i}\boldsymbol{x}_i or_\perp, x_{i+1}) \\
&\vdots \text{ (continue unrolling for } d-i-1\text{-steps)} \\
&= \lambda^{d-i} \max_{x_{i+1}} \ldots \max_{x_d} \breve{Q}^*(\tau\overline{\boldsymbol{xor}}_{\perp<i}\boldsymbol{x}_i or_\perp x_{i+1} or_\perp \ldots x_{d-1} or_\perp, x_d) \\
&\overset{(c)}{=} \lambda^{d-i} \max_{a \in \mathscr{A}(\boldsymbol{x}_{\leq i})} Q^*(h, a) \qquad (14)
\end{aligned}$$

where, again (a) is Equation (8) with the optimal policy, (b) follows from the definition of $\breve{P}$ and $r_\perp = 0$, and (c) is true from the fact that the claim holds for $i = d$ and the maximum is over the restrictive set of actions. □

What we have proven so far is that the sequentialization scheme produces action-value functions which (at the "partial" histories) are rescaled versions of the original action-value function. They agree with the original Q^* at the decision points (at the "complete" histories) where the sequentialized policy $\breve{\Pi}$ completes an action code.

We also show that a similar relationship as proved in Lemma 4.4 holds for a fixed policy $\breve{\Pi}$. However, we use a different proof method for the following lemma. Note that $\breve{\Pi}$ induces a policy $\bar{\Pi}$ on the original environment, which can trivially be expressed as follows:

$$\bar{\Pi}(a|h) := \prod_{i=1}^{d} \breve{\Pi}(\boldsymbol{x}_i|\tau\overline{\boldsymbol{xor}}_{\perp<i}) =: \breve{\Pi}(\boldsymbol{x}|\tau) \qquad (15)$$

for any $a = D(\boldsymbol{x})$ and $\tau = g(h)$.

Lemma 4.5 ($\breve{Q}^{\breve{\Pi}}$ $\boldsymbol{x}$-relationship)**.** *For any arbitrary policy $\breve{\Pi}$ the following relationship is true:*

$$\breve{Q}^{\breve{\Pi}}(\tau\overline{\boldsymbol{xor}}_{\perp<d}, \boldsymbol{x}_d) = Q^{\bar{\Pi}}(h, D(\boldsymbol{x})) \qquad (16)$$

for any history $\tau = g(h)$ and $\boldsymbol{x} \in \mathscr{B}^d$.

Proof. Before we prove the main result of the lemma, we show that the following relationship holds for the value-functions of the sequentialized and the original environment:

$$V^{\breve{\Pi}}(\tau) = \lambda^{d-1} V^{\bar{\Pi}}(h) \qquad (17)$$

for any $\tau = g(h)$. We use a different argument than Lemma 4.4 to prove the above statement. Lets imagine the sequentialized environment is at the history $\tau = g(h)$. The agent starts to follow the policy $\breve{\Pi}$. The following is the (expected-reward, discount-factor) sequence it generates from this history.

$$\begin{aligned}
&(0, \lambda^0), (0, \lambda^1), \ldots, (0, \lambda^{d-2}), (\bar{r}, \lambda^{d-1}), \\
&(0, \lambda^d), (0, \lambda^{d+1}), \ldots, (0, \lambda^{2d-2}), (\bar{r}', \lambda^{2d-1}), \\
&(0, \lambda^{2d}), \ldots
\end{aligned}$$

where $\bar{r}$ is the expected reward. The sum of the reward part of the above sequence returns $\breve{V}^{\breve{\Pi}}(\tau)$. Now, if we re-scale the discount part of the above sequence by λ^{d-1} we get $V^{\bar{\Pi}}(h)$ as the sum of the reward part.

$$
\begin{aligned}
&(0, \lambda^{1-d}), (0, \lambda^{2-d}), \dots, (0, \lambda^{-1}), (\bar{r}, \lambda^{0}),\\
&(0, \lambda^{1}), (0, \lambda^{2}), \dots, (0, \lambda^{d-1}), (\bar{r}', \lambda^{d}),\\
&(0, \lambda^{d+1}), \dots
\end{aligned}
$$

which proves Equation (17) when $\lambda^d = \gamma$. Now, let $a := D(\boldsymbol{x})$.

$$
\begin{aligned}
Q^{\breve{\Pi}}(\tau\overline{\boldsymbol{xor}}_{\perp<d}, \boldsymbol{x}_d) &= \sum_{o'r'} \breve{P}(o'r'|\tau\overline{\boldsymbol{xor}}_{\perp<d}\boldsymbol{x}_d) \left(r' + \lambda V^{\breve{\Pi}}(\tau\overline{\boldsymbol{xor}}_{\perp<d}\boldsymbol{x}_d o'r')\right)\\
&\overset{(a)}{=} \sum_{o'r'} P(o'r'|ha) \left(r' + \lambda V^{\breve{\Pi}}(\tau\overline{\boldsymbol{xor}}_{\perp<d}\boldsymbol{x}_d o'r')\right)\\
&\overset{(17)}{=} \sum_{o'r'} P(o'r'|ha) \left(r' + \lambda^d V^{\bar{\Pi}}(hao'r')\right)\\
&= \sum_{o'r'} P(o'r'|ha) \left(r' + \gamma V^{\bar{\Pi}}(hao'r')\right) = Q^{\bar{\Pi}}(h, D(\boldsymbol{x}))
\end{aligned}
$$

where (a) is due to Proposition 4.1. □

The following theorem proves the usefulness of our sequentialization framework. We show that the optimal policy of the sequentialized environment is also optimal in the original environment when it is lifted back using the decoding function D.

Theorem 4.6 (Sequentialization preserves ε-optimality)**.** *Any $\lambda^{d-1}\varepsilon$-optimal policy of the sequentialized environment is ε-optimal in the original environment.*

Proof. Let $\breve{\Pi}$ be an ε'-optimal policy of the sequentialized environment, where $\varepsilon' := \lambda^{d-1}\varepsilon$. It implies the following:

$$
\breve{V}^{*}(\tau\overline{\boldsymbol{xor}}_{\perp<i}) - \breve{V}^{\breve{\Pi}}(\tau\overline{\boldsymbol{xor}}_{\perp<i}) \leq \varepsilon' \tag{18}
$$

for any complete sequentialized history $\tau = g(h)$ and $\boldsymbol{x} \in \mathscr{B}^{i-1}$ where $i \leq d$. Especially, we are interested in the case when $i = 1$, i.e. values at the complete histories.

$$
\breve{V}^{*}(\tau) - \breve{V}^{\breve{\Pi}}(\tau) \leq \varepsilon' \tag{19}
$$

With simple algebra, we can show that the following relationship holds for the optimal policies of the sequentialized and original processes:

$$
\begin{aligned}
\breve{V}^{*}(\tau) &\overset{(a)}{=} \max_{x} \breve{Q}^{*}(\tau, x)\\
&\overset{(b)}{=} \lambda^{d-1} \max_{\boldsymbol{x}\in\mathscr{B}^d} \breve{Q}^{*}(\tau\overline{\boldsymbol{xor}}_{\perp<d}, \boldsymbol{x}_d)\\
&\overset{(c)}{=} \lambda^{d-1} \max_{\boldsymbol{x}\in\mathscr{B}^d} Q^{*}(h, D(\boldsymbol{x})) = \lambda^{d-1} V^{*}(h)
\end{aligned} \tag{20}
$$

where (a) is the definition of the value function, (b) holds due to Proposition 4.3, and (c) is true by applying Lemma 4.4 for $i = d$.

Now, by simply using Equation (17) and Equation (20), we can prove the claim.

$$V^*(h) - V^{\bar{\Pi}}(h) \overset{(a)}{=} \lambda^{1-d}\left(\breve{V}^*(\tau) - \breve{V}^{\breve{\Pi}}(\tau)\right) \overset{(19)}{\leq} \varepsilon \tag{21}$$

for any $\tau = g(h)$, where (a) is due to Equation (17) and Equation (20). □

We are done formally defining the setup. In the next section we put everything together under the context of ESA to establish the validity of our sequentialization setup.

5 Extreme State Aggregation

In the previous sections, we formalized the GRL problem with a sequentialized action-space. A GRL agent keeps the history of its interaction to decide the next action. The history grows with time but even worse is that, without more assumptions and/or abstractions, no history ever repeats [**?**]. This is a unique characteristic of the history-based setup which sets it apart from the standard RL [**?**]. It enables the GRL framework to cover from the extreme case of unique histories to the most restrictive scenarios of bandits. However, without abstractions, a GRL agent which assumes every history is unique is more of a theoretical artifact than a realizable algorithm. It is critical to note that our sequentialization scheme results in just like any other GRL agent. It also requires an abstraction map (or further structural assumptions) to provide an implementable algorithm. Usually, one starts by assuming some structure on the history set(s). After reviewing some, we will argue against all of them.

On one extreme we have unique histories and on the other end, typically, the environment distribution is assumed to be Markovian, i.e. for any h and a, $P(o'r'|ha) \equiv P(o'r'|oa)$ for all $o'r'$ where o is the most recent observation in h [**?**]. This means that histories with the same most recent observation are members of the same class (or state). This assumption provides a lot of structure on $\mathscr{H}$. The value functions become functions of the most recent observations. Note that **?**] defines the Markovian assumption directly on histories, which is a bit weaker than what we have stated above.

Unfortunately, the Markovian assumption is too strong to be used in many real-world problems. We do not interact with the world based on just our recent observations. As general agents, we keep "relevant" historical events in memory to plan better in the future, sometimes optimally. Apart from some "toy" examples and (well-defined) games [**?** **?** **?**], this assumption demands too much structure on the history space. So, what other assumption can we make? We can keep the Markovian structure but can weaken the assumption, significantly, by assuming that the agent is not able to observe the state directly. The agent may require a sufficiently long history of interaction to discern the *hidden* Markovian state of the environment [**?**]. The class of problems this assumption models is known as partially observable Markov decision problems (POMDP). Almost all problems we care about can be modeled as POMDPs. However, POMDP solution methods are very demanding and the optimal behavior is not guaranteed to be learnable in general [**?**]. We do not address this non-MDP class any further.

We focus on other important quantities in GRL formulation, e.g. Π^*, V^*, and Q^*, and make no direction assumption on P. This has been a subject of many works; [**?** **?**] considered a unified abstraction framework by mapping states similar in value, **?**] subsumed the previous work by considering the GRL setup, and **?**] extended the work to state-action abstractions. The value functions provide natural criteria to group histories. The resultant structure can be non-Markovian. Such non-MDP abstractions have many benefits over Markovian reductions.

The resultant state-space ($\cong$ the set of groups of histories) can be significantly smaller with these abstractions than the Markovian counterparts. One advantage of using such abstractions, as compared to POMDPs, is the guarantee of the optimal behavior being a function of states, which helps the learning in many problems that were traditionally not considered learnable [**?**].

However, the most remarkable aspect of such non-MDP abstractions is that there may exist an upper bound on the required number of states *uniformly* for any problem, as it is the case in ESA [**?**]. The idea is to group histories together which have *similar* optimal action-values Q^*. Since Q^* is a bounded real function (which is the case as $\mathscr{R}$ is bounded), we can potentially upper bound the required number of states by lumping together histories by discretization of the action-value function. In this work, we are primarily interested in non-MDP abstractions of the following type.

Definition 5.1 (ε-Q-uniform abstraction)**.** *An abstraction function $\phi : \mathscr{H} \to \mathscr{S}$ is an ε-Q-uniform abstraction if for any $h, \dot{h} \in \mathscr{H}$ and all $a \in \mathscr{A}$ we have*

$$\Big(\phi(h) = \phi(\dot{h})\Big) \implies \Big|Q^*(h,a) - Q^*(\dot{h},a)\Big| \leq \varepsilon$$

where $\mathscr{S}$ is the set of states[9] *of the abstraction.*

In ESA, the agent's policy is (constrained to be only) a function of the states. Although these states do not exhibit Markovian dynamics, the agent can "pretend" that the abstract process is Markovian. This structure provides a surrogate-MDP whose optimal policy is ε-optimal in the original environment.

The ε-Q-uniform, non-MDP abstractions lead to the following important result due to **?**]. We only state the result without a proof for the closure of exposition, see **?**] for more details about ESA and proofs.

In the following theorems we assume that the rewards are bounded in the unit interval, i.e. $\mathscr{R} \subseteq [0,1]$. This is done for brevity, and it is not a necessary condition. The rescaling of the rewards does not affect the decision-making process in (G)RL. In general, let the range of the rewards be $R := \max \mathscr{R} - \min \mathscr{R}$. Then, the scalars in the nominators of Theorems 5.2 and 5.3 are replaced by $2R$ and $4R^2$ respectively.

Theorem 5.2 (ESA [**?** , Theorem 11])**.** *For every environment P there exists a reduction ϕ and a surrogate-MDP whose optimal policy*[10] *is an ε-optimal policy for the environment. The size of the surrogate-MDP is bounded (uniformly for any P) by*[11]

$$|\mathscr{S}| \leq \left(\frac{2}{\varepsilon(1-\gamma)^3}\right)^{|\mathscr{A}|}$$

This is a powerful result, but it suffers from the exponential dependence on the action-space size. We now put our action sequentialization framework to work and dramatically improve this dependency from exponential to only a logarithmic dependency in $|\mathscr{A}|$.

So far, we have considered an arbitrary $\mathscr{B}$-ary decision set to sequentialize the action-space. However, in the following theorem we go to the extreme case of sequentializing the

[9] We consider a finite abstract set of states, but the underlying set of states ($\cong$ history-space) is (allowed to be) infinite. Note also that we consider *approximate* Q-uniform aggregations which result into a finite abstract state.

[10] See **?**] of how to learn this policy, the surrogate-MDP, Q^*, and ϕ.

[11] The 2 instead of a 3 in the original theorem is a trivial improvement by removing the grid point at 0 in the construction.

action-space to binary decisions ($\mathscr{B} = \mathbb{B}$) to squeeze out the maximum improvement possible through the framework.

Theorem 5.3 (Binary ESA). *For every environment there exists an abstraction and a corresponding surrogate-MDP for its binarized version ($\mathscr{B} = \mathbb{B}$) whose optimal policy is ε-optimal for the true environment. The size of the surrogate-MDP is uniformly bounded for* every *environment as*

$$|\mathscr{S}| \leq \frac{4\lceil 1 - \gamma + \log_2|\mathscr{A}|\rceil^6}{\gamma^2\varepsilon^2(1-\gamma)^6}$$

Proof. Consider the agent that is interacting with the sequentialized/binarized environment $\breve{P}$. By Theorem 4.6, we know that a near-optimal policy of this sequentialized environment is also near-optimal in the original environment. Now, if we use ESA on the binarized problem and get an ε'-optimal policy through the surrogate-MDP by Theorem 5.2, we are sured to be ε-optimal in the original environment P as explained above. Additionally, the size of the state-space is bounded as

$$|\mathscr{S}| \overset{\textit{Theorem 5.2}}{\leq} \left(\frac{2}{\varepsilon'(1-\lambda)^3}\right)^2 = \frac{4}{\varepsilon'^2(1-\lambda)^6} \tag{22}$$

where λ is the discount factor of the sequentialized problem. Next, we upper bound Equation (22) by using the fact that $\lambda^d = \gamma$. Let $\delta := 1 - \gamma < 1$. So,

$$\begin{aligned} 1 - \lambda &= 1 - (1-\delta)^{1/d} = 1 - \mathrm{e}^{\frac{\ln(1-\delta)}{d}} \\ &\overset{(a)}{\geq} 1 - \frac{1}{1 - \ln(1-\delta)/d} \overset{(b)}{\geq} 1 - \frac{1}{1+\delta/d} \\ &= \frac{\delta}{d+\delta} = \frac{1-\gamma}{d+1-\gamma} \end{aligned} \tag{23}$$

where (a) holds due to $\frac{1}{\mathrm{e}^{-\alpha}} \leq \frac{1}{1-\alpha}$, (b) is true by using the fact that $\delta < 1$, hence $\ln(1-\delta) \leq -\delta$. Therefore, using Equation (22), Equation (23), and $\varepsilon' = \lambda^{d-1}\varepsilon \geq \lambda^d\varepsilon = \gamma\varepsilon$ we get,

$$|\mathscr{S}| \leq \frac{4}{\varepsilon'^2(1-\lambda)^6} \leq \frac{4(1-\gamma+d)^6}{\gamma^2\varepsilon^2(1-\gamma)^6} \tag{24}$$

which proves the claim. □

Superficially, it might seem that we have simply replaced the original discount factor with a larger value. But, it is not the case. If we simply scaled the discount factor (without sequentializing the actions) then the resulting bound would indeed deteriorate, see Theorem 5.2, but on the contrary, with sequentialization/binarization and our analysis the bound (dramatically) improves.

Usually in RL the discount factor γ is close to 1. In that case, the bound in Theorem 5.3 can be tightened further as:

$$|\mathscr{S}| \lesssim \frac{4\lceil \log_2|\mathscr{A}|\rceil^6}{\varepsilon^2(1-\gamma)^6} \tag{25}$$

which agrees with the bound in Theorem 5.2 for the case when $|\mathscr{A}| = 2$, i.e. when the original problem already has a binary action-space.

6 Conclusion & Outlook

This work contributes to the study of the GRL problem. We have provided a reduction to handle large state and action spaces by sequentializing the decision-making process. This helped us improve the upper bound on the number of states in ESA from an exponential dependency in $|\mathscr{A}|$ to logarithmic. The gain is *double exponential* in terms of the action-space dependence at no other cost.

Our result carries a broader impact on the implementation of *general* RL agents[12]. The required storage for such agents, which have access to a non-MDP, approximate Q-uniform abstraction, can be reasonably bounded which only scales logarithmically in the size of the action-space.

We conclude the paper with some future research directions. This work analyses the case when the agent has a fixed aggregation map. **?**] provides an outline for a learning algorithm to learn such abstractions which can be combined with our sequentialization framework.

Another direction, which we also did not touch in this work, is to explore the connection, if any, between the surrogate-MDPs of a map on the original environment, and its extension on the sequentialized problem. By lifting the small binary ESA map, say ψ, back to $\mathscr{H}$, one obtains a small map directly on $\mathscr{H}$, say ϕ. While ψ used sequentialization/binarization for the construction of ϕ, the map ϕ can be used without further referencing to sequentialization. This suggests that a bound logarithmic in $|\mathscr{A}|$ should be possible without a detour through the sequentialization. This deserves further investigation.

We sequentialize the action-space through an *arbitrary* coding scheme C, so the main result does not depend on this choice. Sometimes, it is possible that the action-space may allow natural sequentialization, e.g. in a video game controller the macro action might be a binary vector where the first bit might represent the left/right direction, the second bit indicates up/down, and so on. The exact nature of these binary decisions depends on the domain which is reflected by the choice of encoding C. Sequentialization was our path to double-exponentially improve that bound. Whether there are more direct/natural aggregations with the same bound is an open problem. Moreover, if the agent is learning an abstraction through interaction, the choice of these functions may become critical.

This paper focused on rigorously formalizing and proving the main improvement result. One can also try to empirically show the effectiveness of our improved upper bound. To do this, we need a problem domain where ESA requires more states than the sequentialized/binarized version of it. But a point of caution is that the upper bound still scales badly in terms of γ and ε. Any reasonable value of these parameters would imply a huge upper bound. Even with Markovian abstractions, a cubic dependency on the discount factor is the best achievable. We considered a general underlying process and non-Markovian abstractions, and dramatically improved the previously best bound $(1-\gamma)^{-3|\mathscr{A}|}$ to $(1-\gamma)^{-3\cdot 2}$. Indeed it would be interesting to see whether this can be further improved to the optimal $(1-\gamma)^{-3}$ rate.

Acknowledgements. This work has been supported by Australian Research Council grant DP150104590. Thanks to the anonymous reviewers for their feedback and to Andrs Gyrgy who pointed out that the sequentialized/binarized process in Figure 2 preserves the Markov property, which encouraged us to also consider the Markov case.

Esse fugiat asperiores veniam tenetur similique praesentium facere, mollitia repellat dolorem maiores atque tenetur porro, ab ea voluptas eius corrupti atque nesciunt ex, animi atque eum officiis suscipit voluptatum molestiae, mollitia in praesentium doloribus modi hic molestias debitis inventore

[12]The general (or strong) agents are designed to work with a wide range of environments [**?**].

sint perferendis.Iure voluptas minus eligendi quibusdam incidunt porro quia iste corporis voluptates, in recusandae veniam autem odio perspiciatis consequuntur officia voluptatem, earum recusandae harum eius officia?