On the Limitations of Markovian Rewards to Express Multi-Objective, Risk-Sensitive, and Modal Tasks (Supplementary Material)

Joar Skalse[1,2] **Alessandro Abate**[1]

[1]Computer Science Department, Oxford University, Oxford, UK
[2]The Future of Humanity Institute, Oxford, UK,

A PROOFS

Here, we will provide all proofs that were omitted from the main text. We will begin with Theorem 1, from Section 3.

Theorem 1. *If a MOMDP $\mathcal{M}$ with objective $\mathcal{O}$ is scalarizable, then there exist $w_1 \dots w_k \in \mathbb{R}$ such that $\mathcal{M}$ with $\mathcal{O}$ is scalarized by the reward $R(s,a) = \sum_{i=1}^{k} w_i \cdot R_i(s,a)$.*

To prove this, we must first set up some theoretical preliminaries. For convenience, let $n = |S||A|$, let $T = \mathcal{S} \times \mathcal{A}$, and let each transition in $\mathcal{S} \times \mathcal{A}$ be indexed by an integer $i \in [1, n]$. Moreover, given a reward function R, let $\vec{R} \in \mathbb{R}^n$ be the vector such that $\vec{R}_i = R(T_i)$. Next, given τ, μ_0, and γ, let $m_{\tau,\mu_0,\gamma} : \Pi \to \mathbb{R}^n$ be the function where

$$m_{\tau,\mu_0,\gamma}(\pi)_i = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\xi \sim \pi}(\xi = T_i).$$

Now $J(\pi) = \vec{R} \cdot m_{\tau,\mu_0,\gamma}(\pi)$. In other words, this construction lets us decompose J into two steps, the first of which embeds π in $\mathbb{R}^n$, and the second of which is a linear function. Let S_γ be the smallest affine subspace of R^n such that $\mathrm{Im}(m_{\tau,\mu_0,\gamma}) \in S_\gamma$. We will also use the following lemma:

Lemma 1. $\mathrm{Im}(m_{\tau,\mu_0,\gamma})$ *is open in* S_γ.

For a proof of Lemma 1, see Skalse and Abate [2023] (their Lemma A.11). We can now prove Theorem 1:

Proof. Suppose the MOMDP $\langle \mathcal{S}, \mathcal{A}, \tau, \mu_0, \mathbf{R}, \gamma \rangle$ with $\mathcal{O}$ is equivalent to the MDP $\langle \mathcal{S}, \mathcal{A}, \tau, \mu_0, R, \gamma \rangle$.

First, note that $J(\pi) = \vec{R} \circ m_{\tau,\mu_0,\gamma}(\pi)$, and that $J_i(\pi) = \vec{R}_i \circ m_{\tau,\mu_0,\gamma}(\pi)$ for each of $R_i \in \mathbf{R}$. Let M be the $(n \times k)$-dimensional matrix that maps each vector $x \in \mathbb{R}^n$ to $\langle R_1 \cdot x, \dots, R_k \cdot x \rangle$. In other words, M is the matrix whose rows are $\vec{R}_1 \dots \vec{R}_k$. Since $J(\pi)$ is a function of $J_1(\pi) \dots J_k(\pi)$, we have that $\vec{R} \cdot x_1 = \vec{R} \cdot x_2$ if $M \cdot x_1 = M \cdot x_2$ for any $x_1, x_2 \in \mathrm{Im}(m_{\tau,\mu_0,\gamma})$.

We will first show that $\vec{R} \cdot x_1 = \vec{R} \cdot x_2$ if $M \cdot x_1 = M \cdot x_2$ for any $x_1, x_2 \in S_\gamma$, not just any $x_1, x_2 \in \mathrm{Im}(m_{\tau,\mu_0,\gamma})$. Let x_1, x_2 be any two points in S_γ such that $M \cdot x_1 = M \cdot x_2$, and let x be some arbitrary element of $\mathrm{Im}(m_{\tau,\mu_0,\gamma})$. Let $y_1 = x_1 - x$ and $y_2 = x_2 - x$. Since $\mathrm{Im}(m_{\tau,\mu_0,\gamma})$ is open in S_γ (as per Lemma 1), there is an $\alpha > 0$ such that $x + \alpha \cdot y_1 \in \mathrm{Im}(m_{\tau,\mu_0,\gamma})$ and $x + \alpha \cdot y_2 \in \mathrm{Im}(m_{\tau,\mu_0,\gamma})$. Since M is linear, and since $M \cdot x_1 = M \cdot x_2$, we have that $M \cdot (x + \alpha \cdot y_1) = M \cdot (x + \alpha \cdot y_2)$. Moreover, since $x + \alpha \cdot y_1 \in \mathrm{Im}(m_{\tau,\mu_0,\gamma})$ and $x + \alpha \cdot y_2 \in \mathrm{Im}(m_{\tau,\mu_0,\gamma})$, this means that $\vec{R} \cdot (x + \alpha \cdot y_1) = \vec{R} \cdot (x + \alpha \cdot y_2)$. Finally, from the properties of linear functions, this in turn implies that $\vec{R} \cdot x_1 = \vec{R} \cdot x_2$. Thus, if $M \cdot x_1 = M \cdot x_2$ then $\vec{R} \cdot x_1 = \vec{R} \cdot x_2$ for all $x_1, x_2 \in S_\gamma$.

Next, note that we can decompose M into two matrices M_1, M_2 such that $M = M_1 \cdot M_2$, where M_1 is invertible, and M_2 is an orthogonal projection such that $M_2(x_1) = M_2(x_2)$ if and only if $M(x_1) = M(x_2)$. This means that $\vec{R} \cdot x = \vec{R} \cdot M_2(x)$ for all $x \in S_\gamma$. From this, we obtain that $\vec{R} \cdot x = \vec{R} \cdot M_1^{-1} \cdot M_1 \cdot M_2(x) = \vec{R} \cdot M_1^{-1} \cdot M(x)$ for all $x \in S_\gamma$. Since $\vec{R} \cdot M_1^{-1}$ is a linear function, this means that $\vec{R} \cdot x$ can be expressed as $\sum_{i=1}^{k} w_i \cdot M(x)_i$ for some $w_1 \dots w_k$ for all $x \in S_\gamma$.

Recall that $J(\pi) = \vec{R} \cdot m_{\tau,\mu_0,\gamma}(\pi)$, where $m(\pi) \in S_\gamma$. This means that $J(\pi) = \sum_{i=1}^{k} w_i \cdot M(m_{\tau,\mu_0,\gamma}(\pi))_i = \sum_{i=1}^{k} w_i \cdot \vec{R}_i \cdot m_{\tau,\mu_0,\gamma}(\pi) = \sum_{i=1}^{k} w_i \cdot J_i(\pi)$. This completes the proof. □

Corollary 1. *If $\mathcal{O}(J_1 \dots J_k)$ has a non-linear representation U, and $\mathcal{M}$ is a MOMDP whose J-functions are $J_1 \dots J_k$, then $\mathcal{M}$ with $\mathcal{O}$ is not equivalent to any MDP.*

Proof. Assume for contradiction that $\mathcal{M}$ with $\mathcal{O}$ is equivalent the MDP $\langle \mathcal{S}, \mathcal{A}, \tau, \mu_0, R, \gamma \rangle$. Then J represents $\mathcal{O}(J_1 \dots J_k)$, and this in turn means that U must be strictly monotonic in J. Moreover, Theorem 1 implies that $J = \sum_{i=0}^{k} w_i \cdot J_i$ for some $w_1 \dots w_k \in \mathbb{R}^k$. However, this contradicts our assumptions. □

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

Corollary 2. *There is no MDP equivalent to $\mathcal{M}$ with **LexMax**, as long as $\mathcal{M}$ has at least two reward functions that are neither trivial, equivalent, or opposite.*

Proof. Suppose $\mathcal{M}$ with `LexMax` is equivalent to $\tilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \tau, \mu_0, \tilde{R}, \gamma \rangle$. Let i be the smallest number such that R_i is non-trivial, and let j be the smallest number greater than i such that R_j is non-trivial, and not equivalent to or opposite of R_i. Then there are π_1, π_2 such that $J_i(\pi_1) = J_i(\pi_2)$ and $J_j(\pi_1) < J_j(\pi_2)$, which means that $\pi_1 \prec^{\mathcal{M}}_{\texttt{Lex}} \pi_2$. Moreover, since $\tilde{J}$ represents $\prec^{\mathcal{M}}_{\texttt{Lex}}$, it follows that there are no π, π' such that $J_i(\pi) < J_i(\pi')$ and $\tilde{J}(\pi) > \tilde{J}(\pi')$. Then Theorem 1 in Skalse et al. [2022] implies that R_i is equivalent to $\tilde{R}$. However, then $\tilde{J}(\pi_1) = \tilde{J}(\pi_2)$, which means that $\tilde{J}$ cannot represent $\prec^{\mathcal{M}}_{\texttt{Lex}}$. □

Corollary 3. *There is no MDP equivalent to $\mathcal{M}$ with **MaxMin**, unless $\mathcal{M}$ has a reward function R_i such that $J_i(\pi) \leq J_j(\pi)$ for all $j \in \{1 \dots k\}$ and all π.*

Proof. $\mathcal{O}^{\mathcal{M}}_{\texttt{Min}}$ is represented by the function $U(\pi) = \min_i J_i(\pi)$. Moreover, if $\mathcal{M}$ has no reward function R_i such that $J_i(\pi) \leq J_j(\pi)$ for all $j \in \{1 \dots k\}$ and all π then this representation is non-linear. Corollary 1 then implies that $\mathcal{M}$ with `MaxMin` is not equivalent to any MDP. □

Corollary 4. *There is no MDP equivalent to $\mathcal{M}$ with **MaxSat**, as long as $\mathcal{M}$ has at least one reward R_i where $J_i(\pi_1) < c_i$ and $J_i(\pi_2) \geq c_i$ for some $\pi_1, \pi_2 \in \Pi$.*

Proof. Note that $\texttt{MaxSat}(\mathcal{M})$ is represented by the function $U(\pi) = \sum_{i=1}^{k} \mathbb{1}[J_i(\pi) \geq c_i]$, where $\mathbb{1}[J_i(\pi) \geq c_i]$ is the function that is equal to 1 when $J_i(\pi) \geq c_i$, and 0 otherwise. Moreover, U is not strictly monotonic in any function that is linear in $J_1 \dots J_k$. Corollary 1 thus implies that $\mathcal{M}$ with `MaxSat` is not equivalent to any MDP. □

Corollary 5. *There is no MDP equivalent to $\mathcal{M}$ with **ConSat**, unless either R_1 and R_2 are equivalent, or $\max_\pi J_1(\pi) \leq c$.*

Proof. $\mathcal{O}^{\mathcal{M}}_{\texttt{Con}}$ is represented by $U(\pi) = \{J_1(\pi)$ if $J_1(\pi) \leq c$, else $J_2(\pi) - \min_\pi J_2(\pi) + c\}$. Moreover, this representation is non-linear, unless either R_1 and R_2 are equivalent, or $\max_\pi J_1(\pi) \leq c$. Corollary 1 then implies that $\mathcal{M}$ with `ConSat` is not equivalent to any MDP. □

We next give the proof of Theorem 2, from Section 4.

Theorem 2. *Given $\mathcal{S}$, $\mathcal{A}$, and γ, let R_1 and R_2 be two reward functions. If for all $\xi_1, \xi_2 \in (\mathcal{S} \times \mathcal{A})^\omega$ and $\gamma \geq 0.5$,*

$$G_1(\xi_1) \leq G_1(\xi_2) \iff G_2(\xi_1) \leq G_2(\xi_2),$$

then there exist $a \in \mathbb{R}$, $b \in \mathbb{R} > 0$ such that for all $\xi \in (\mathcal{S} \times \mathcal{A})^\omega$,

$$G_1(\xi) = b \cdot G_2(\xi) + a.$$

Proof. We can first note that if G_1 is constant then G_2 must also be constant, and vice versa, in which case this result is straightforward (with $b = 1$, $a = G_1 - G_2$). For the rest of the proof, assume that neither G_1 or G_2 is constant.

For convenience, let $n = |S||A|$, let $T = \mathcal{S} \times \mathcal{A}$, and let each transition in $\mathcal{S} \times \mathcal{A}$ be indexed by an integer $i \in [1, n]$. Let $\vec{R_1} \in \mathbb{R}^n$ be the vector such that $\vec{R_1}_i = R_1(T_i)$, and $\vec{R_2} \in \mathbb{R}^n$ be the vector such that $\vec{R_2}_i = R_2(T_i)$. Moreover, let $m : T \to \mathbb{R}^n$ be the function where

$$m(\xi)_i = \sum_{j=0}^{\infty} \delta^j \mathbb{1}[\xi_j = T_i].$$

Now $G_1(\xi) = \vec{R_1} \cdot m(\xi)$ and $G_2(\xi) = \vec{R_2} \cdot m(\xi)$. In other words, this construction lets us decompose G_1 and G_2 into two steps, the first of which embeds ξ in $\mathbb{R}^n$, and the second of which is a linear function.

Next, let us consider what $\mathrm{Im}(m)$ looks like. First, note that $m(\xi)_i \geq 0$ for all i and all ξ. Next, note that $\sum m(\xi) = 1/(1-\gamma)$ for all ξ. This means that $\mathrm{Im}(m)$ is located inside the simplex that is formed by all points in the positive quadrant of $\mathbb{R}^n$ whose L_1-norm is $1/(1-\gamma)$.

Consider two arbitrary transitions $t_i, t_j \in T$. Note that $m(t_i^\omega)$ is the point where the aforementioned simplex intersects the i'th basis vector of $\mathbb{R}^n$, and similarly for $m(t_j^\omega)$. Moreover, if ξ is made up entirely from t_i and t_j in some combination and order (i.e., $\xi \in \{t_i, t_j\}^\omega \subseteq T$), then $m(\xi)$ is on the line between $m(t_i^\omega)$ and $m(t_j^\omega)$.

Let α be any number in $[0, 1/(1-\gamma)]$. Since $1/\gamma > 1$, there is a representation of α in base $1/\gamma$. This means that there is an integer u and a sequence of integers $\{a_k\}_{k \in (-\infty, u]}$ such that

$$\sum_{k=u}^{-\infty} a_k \cdot (1/\gamma)^k = \alpha$$

where each a_k is a nonnegative integer less than $1/\gamma$. Since $\gamma \geq 0.5$, this means that each a_k is 0 or 1. Moreover, since $\alpha \leq 1/(1-\gamma)$, we have that $u \leq 0$. By rewriting using $k' = -k$, this means that there is a sequence $\{a_{k'}\}_{k' \in [0, \infty)}$ where each $a_{k'} \in \{0, 1\}$ such that

$$\sum_{k'=0}^{\infty} a_{k'} \cdot \gamma^{k'} = \alpha.$$

Let $\xi \in T$ be the trajectory where $\xi_{k'} = t_i$ if $a_{k'} = 1$, and t_j if $a_{k'} = 0$. We now have that $m(\xi) = \alpha/(1/(1-\gamma)) \cdot m(t_i^\omega) + (1 - \alpha/(1/(1-\gamma))) \cdot m(t_j^\omega)$. Since α was chosen arbitrarily from $[0, 1/(1-\gamma)]$, this means that every point on the line between $m(t_i^\omega)$ and $m(t_j^\omega)$ are in $\mathrm{Im}(m)$. Since t_i and t_j were also chosen arbitrarily, this holds for any t_i and t_j in T.

Consider again the simplex that is formed by all points in the positive quadrant of $\mathbb{R}^n$ whose L_1-norm is $1/(1-\gamma)$.

We have just shown that every point on the edges (1-faces) of this simplex are in $\mathrm{Im}(m)$.

Consider the linear functions that $\vec{R_1}$ and $\vec{R_2}$ induce on $\mathbb{R}^n$. Take the point x at the centre of the simplex, and consider the tangent plane of $\vec{R_1}$ at this point. Since every point on any of the simplex edges are in $\mathrm{Im}(m)$, we have that this tangent plane must intersect $\mathrm{Im}(m)$ at $n-1$ linearly independent points. Since $\vec{R_1}\cdot x_1 = \vec{R_1}\cdot x_2$ implies that $\vec{R_2}\cdot x_1 = \vec{R_2}\cdot x_2$ for all $x_1, x_2 \in \mathrm{Im}(m)$, we have that the tangent plane of $\vec{R_2}$ at x must intersect $\mathrm{Im}(m)$ at the same points. This implies that there are $a, b \in \mathbb{R}$ such that $G_1 = b \cdot G_2 + a$. Since moreover $G_1(\xi_1) \leq G_1(\xi_2) \iff G_2(\xi_1) \leq G_2(\xi_2)$, we have that $b > 0$. □

Theorem 3. *For any modal reward $R^\Diamond$ and any transition function τ, there exists a reward R that is contingently equivalent to $R^\Diamond$ given τ. Moreover, unless $R^\Diamond$ is trivial, there is no reward that is robustly equivalent to $R^\Diamond$.*

Proof. This is straightforward. For the first part, simply let $R(s,a,s') = R^\Diamond(s,a,s',\tau)$. The second part is immediate from the definition of trivial modal reward functions. □

B TASKS AS OPTIMAL POLICIES

In this paper, we primarily think of a "task" as corresponding to a policy ordering. An alternative way to formalise the notion of a task is as a set of optimal policies. It is fairly straightforward to provide necessary and sufficient conditions for when this type of task can be expressed using a scalar, Markovian reward function.

Proposition 1. *A set of policies $\hat{\Pi}$ is the optimal policy set for some reward if and only if there is a function $o : \mathcal{S} \to \mathcal{P}(\mathcal{A}) \setminus \varnothing$ that maps each state to a (non-empty) set of "optimal actions", and $\pi \in \hat{\Pi}$ if and only if $\mathrm{supp}(\pi(s)) \subseteq o(s)$.*

Proof. For the "if" part, consider the reward function R where $R(s,a,s') = 0$ if $a \in o(s)$, and $R(s,a,s') = -1$ otherwise. The "only if" part follows from the fact that the optimal Q-function $Q^\star$ is the same for all optimal policies, so we can let $o(s) = \mathrm{argmax}_a Q^\star(s,a)$. □

We can see that some tasks of this form cannot be expressed by Markovian rewards. For example, consider the task "always go in the same direction" — this task cannot be expressed as a reward function, because any policy that mixes the actions of two other optimal policies must itself be optimal. It also shows that Markovian reward functions cannot be used to encourage *stochastic* policies. For example, there is no Markovian reward function under which "play rock, paper, and scissors with equal probability" is the unique optimal policy.

C MORE MORL OBJECTIVES

In this Appendix, we give even more examples of MORL objectives, and some comments on how to construct them – the purpose of this is mainly just to show how rich this space is. First, similar to the MaxMin objective, we might want to judge a policy according to its *best* performance:

Definition 1. *Given $J_1 \dots J_k$, the **MaxMax** objective $\prec_{\texttt{Max}}$ is given by $\pi_1 \prec_{\texttt{Max}} \pi_2 \iff \max_i J_i(\pi_1) < \max_i J_i(\pi_2)$.*

We would next like to point out that it is possible to create smooth versions of almost any MORL objective. In Section 6, we outline an approach for learning any continuous, differentiable MORL objective, so this is quite useful. We begin with a soft version of the MaxMax objective:

Definition 2. *Given $J_1 \dots J_k$ and $\alpha > 0$, the **Soft MaxMax** objective $\prec_{\texttt{MaxSoft}}$ is given by*

$$J_{\texttt{MaxSoft}}(\pi) = \left(\sum_{i=1}^{k} J_i(\pi) e^{\alpha J_i(\pi)}\right) \Bigg/ \left(\sum_{i=1}^{k} e^{\alpha J_i(\pi)}\right).$$

This is of course not the only way to continuously approximate MaxMax, it is just an example of one way of doing it. Here α controls how "sharp" the approximation is – the larger α is, the closer $J_{\texttt{MaxSoft}}$ gets to the sharp max function, and the smaller α is, the closer it gets to the arithmetic mean function (so by varying α, we can continuously interpolate between them). Similarly, we can also create a smooth version of MaxMin:

Definition 3. *Given $J_1 \dots J_k$ and $\alpha > 0$, the **Soft MaxMin** objective $\prec_{\texttt{MinSoft}}$ is given by*

$$J_{\texttt{MinSoft}}(\pi) = \left(\sum_{i=1}^{k} J_i(\pi) e^{-\alpha J_i(\pi)}\right) \Bigg/ \left(\sum_{i=1}^{k} e^{-\alpha J_i(\pi)}\right).$$

As before, the larger α is, the closer $J_{\texttt{MinSoft}}$ gets to the sharp min function, and the smaller α is, the closer it gets to the arithmetic mean function We can also smoothen MaxSat:

Definition 4. *Given $J_1 \dots J_k$, $c_1 \dots c_k$, and $\alpha > 0$, the **Soft MaxSat** objective $\prec_{\texttt{SatSoft}}$ is*

$$J_{\texttt{SatSoft}}(\pi) = \sum_{i=1}^{k} \left(\frac{1}{1 + e^{-\alpha(J_i(\pi) - c_i)}}\right).$$

The larger α is, the closer $J_{\texttt{SatSoft}}$ gets to the sharp MaxSat function (and the smaller α gets, the closer $J_{\texttt{SatSoft}}$ gets to a flat 0.5). And, again, this is of course not the only way to create a smooth version of MaxSat. It is unclear if it is possible to create a smooth version of ConSat without having any prior knowledge of (a lower bound of)

the value of $\min_\pi J_1(\pi)$, but with this value it should be reasonably straightforward (see the construction in Corollary 5). As for LexMax, we can of course create a smooth approximation of it by taking a linear approximation of the weights, but here we would need some prior knowledge of $\max_\pi J_1(\pi) \dots \max_\pi J_k(\pi)$.

D A METHOD FOR SOLVING MODAL TASKS

In this Appendix, we give an outline of one possible method for solving modal tasks. We mainly want to show that it is *feasible* to learn modal tasks, and so we only provide a solution sketch; the task of *implementing* and *evaluating* this method is something we leave as a topic for future work.

We will first define a restricted class of modal tasks, which is both very expressive, and also more amenable to learning than the more general version given in Definition 7:

Definition 5. *An* affordance *consists of a reward function and a discount factor, $\langle R, \gamma \rangle$, and an* affordance-based reward *is a function $R^\Diamond : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R}^{2k} \to \mathbb{R}$, that is continuous in the last $2k$ arguments. An* affordance-based MDP *is a tuple $\langle \mathcal{S}, \mathcal{A}, \tau, \mu_0, R^\Diamond, \gamma, \langle R, \gamma \rangle^k \rangle$, where the reward given for transitioning from s to s' via a is $R^\Diamond(s, a, s', V_1^\star(s) \dots V_k^\star(s), V_1^\star(s') \dots V_k^\star(s'))$, where $V_i^\star$ is the optimal value function of the i'th affordance.*

This definition requires some explanation. In psychology (and other fields, such as user interface design), an affordance is, roughly, a perceived possible action, or a perceived way to use an object. For example, if you see a button, then the fact that you can *press* that button, and expect something to happen, is part of *how you perceive* it, in a way that might not be the case if you could somehow show the button to a premodern human. It can also be used to refer to a choice or action that is perceived as available in some context (without being tied to an object). Here, we are using it to refer to a *task* that could be performed in an MDP. The intuition is that $R^\Diamond$ is allowed to depend on what *could be done* from s and s', in addition to the state features of s and s'.

Before outlining an algorithm, let us first give a few examples of how to formalise modal tasks within this framework. First consider the instruction "you should always be able to return to the start state". We can formalise this using a reward function R_1 that gives 1 reward if the start state is entered, and 0 otherwise, and pair it up with a discount parameter γ that is very close to 1. We could then set $R^\Diamond$ to, for example, $R^\Diamond(s, a, s', V_1^\star(s), V_1^\star(s')) = R(s, a, s') \cdot \tanh(V_1^\star(s'))$, where R describes some base task. In this way, no reward is given if the start state cannot be reached from s'. Next, consider the instruction "never enter a state from which it is possible to quickly enter an unsafe state". To formalise this, let R_1 give 1 reward if an unsafe state is entered, and 0 otherwise, and let γ correspond to a very high discount rate (e.g. 0.7). We could then set $R^\Diamond$ to, for example, $R^\Diamond(s, a, s', V_1^\star(s), V_1^\star(s')) = R(s, a, s') - V_1^\star(s')$, where R again describes some base task.

These examples show that our "affordance-based" MDPs are quite flexible, and that they should be able to formalise many natural modal tasks in a satisfactory way, including most of our motivating examples.[1] However, the definition could of course be made more general. For example, we could allow the affordances to themselves be based on affordance-based reward functions, etc. However, it is not clear if this would bring much benefit in practice.

Let us now outline an approach for solving affordance-based MDPs using reinforcement learning, specifically using an action-value method. First, let the agent maintain $k+1$ Q-functions, $Q^\Diamond, Q_1, \dots, Q_k$, one for $R^\Diamond$ and one for each affordance $\langle R_i, \gamma_i \rangle$. Next, we suppose that the agent updates each of $Q_1, \dots, Q_k$ using an off-policy update rule, such as Q-learning; this will ensure that $Q_1, \dots, Q_k$ converge to their true values (i.e. to $Q_1^\star \dots Q_k^\star$), as long as the agent explores infinitely often. Note that the use of an off-policy update rule is crucial. Next, let the agent update $Q^\Diamond$ as if it were an ordinary Markovian reward function, using the reward $\hat{R}(s, a, s') = R^\Diamond(s, a, s', V_1(s) \dots V_k(s), V_1(s') \dots V_k(s'))$, where $V_i(s)$ is given by $\max_a Q_i(s, a)$. In other words, we let it update $Q^\Diamond$ using an *estimate* of the true value of $R^\Diamond$, expressed in terms of its current estimates of $V_1^\star \dots V_k^\star$. The fact that $Q_1, \dots, Q_k$ converge to $Q_1^\star, \dots, Q_k^\star$, and the fact that $R^\Diamond$ is continuous in its value function arguments, will ensure that the estimate $\hat{R}$ also converges to the true value of $R^\Diamond$. The update rule used for $Q^\Diamond$ could be either on-policy or off-policy. We then suppose that the agent selects its actions by applying a Bandit algorithm to $Q^\Diamond$, and that this Bandit algorithm is greedy in the limit, but also explores infinitely often, as usual.

This algorithm should be able to learn to optimise the reward in any affordance-based MDP. In the tabular case, it should be possible (and reasonably straightforward) to prove that it always converges to an optimal policy (assuming that appropriate learning rates are used, etc), using Lemma 1 in Singh et al. [2000]. We would also expect it to perform well in practice, when used with function approximators (such as neural networks). However, we leave the task of implementing and properly evaluating this approach as a topic for future work.

There are also several ways that this algorithm could be tweaked or improved. For example, the algorithm we have described is an action-value algorithm, but the same approach could of course be used to make an actor-critic algo-

[1] This arguably excludes "you should never enter a state where you would be unable to receive a feedback signal". However, this instruction only makes sense in a multi-agent setting.

rithm instead. We also suspect that there could be interesting modifications one could make to the exploration strategy of the algorithm. If a standard Bandit algorithm (such as ϵ-greedy) is used, then the agent will mostly take actions that are optimal under its current estimate of $Q^{\Diamond}$. In the ordinary case, this is good, because it leads the agent to spend more time in the parts of the MDP that are relevant for maximising the reward. However, in this case, there is a worry that it could lead the agent to neglect the parts of the (affordance-based) MDP that are relevant for learning more about $V_1^{\star} \dots V_k^{\star}$, which might slow down the learning. Again, we leave such developments for future work, since our aim here only is to show that it is feasible to learn non-trivial modal tasks.

We also want to point out that the work by Wang et al. [2020] could provide another starting point for learning modal tasks using RL. In their work, they present some RL-based methods for determining whether a specification in Probabilistic Computational Tree Logic (PCTL) holds in an MDP. PCTL can be used to specify many kinds of properties of states in MDPs which depend on the transition function, including e.g. what states can and cannot be reached from a particular state, and with what probability, etc. We can therefore specify non-trivial modal tasks by providing a number of PCTL formulas, and allowing the reward function to depend on the truth values of these formulas. That is, we could consider a setup that is analogous to that which we give in Definition 5, but where the "affordances" are replaced by PCTL formulas. It should then be possible to learn tasks specified in this manner by using the techniques of Wang et al. [2020] to learn the values of the PCTL formulas, and then using ordinary RL to train on the resulting reward function.

References

Satinder Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38:287–308, 2000.

Joar Skalse and Alessandro Abate. Misspecification in inverse reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Joar Skalse, Niki Howe, Krasheninnikov Dima, and David Krueger. Defining and characterizing reward hacking. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2022.

Yu Wang, Nima Roohi, Matthew West, Mahesh Viswanathan, and Geir E. Dullerud. Statistically model checking pctl specifications on Markov decision processes via reinforcement learning. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1392–1397, 2020. doi: 10.1109/CDC42340.2020.9303982.