A PROOFS OF LEMMAS

In this Appendix, we provide the proofs of the lemmas from Section 3.

Lemma 1. *If R is non-constant, then for any state s there exists trajectories $\zeta_1, \zeta_2, \zeta_3$ starting in s such that $G(\zeta_1) \neq G(\zeta_2)$, $G(\zeta_2) \neq G(\zeta_3)$, and $G(\zeta_1) \neq G(\zeta_3)$.*

Proof. First note that if R is non-constant, then there must be *some* state s and some trajectories ξ_1, ξ_2 starting in s such that $G(\xi_1) \neq G(\xi_2)$ (this follows from Theorem 3.8 in Skalse et al. (2022a)). We will establish that there is a ξ_3 starting in s such that $G(\xi_3) \neq G(\xi_1)$ and $G(\xi_3) \neq G(\xi_2)$, and then show that this implies that such trajectories exist for *all* states.

Suppose for contradiction that for any ξ_3 starting in s, either $G(\xi_3) = G(\xi_1)$ or $G(\xi_3) = G(\xi_2)$. Consider a transition $\langle s, a, s \rangle$, and let $\zeta_1 = \langle s, a, s \rangle + \xi_1$ and $\zeta_2 = \langle s, a, s \rangle + \xi_2$; we will do a case enumeration, and show that either $G(\zeta_1)$ or $G(\zeta_2)$ must be distinct from both $G(\xi_1)$ and $G(\xi_2)$. Note that $G(\zeta_1) = R(s, a, s) + \gamma G(\xi_1)$ and $G(\zeta_2) = R(s, a, s) + \gamma G(\xi_2)$.

Case 1: $G(\zeta_1) = G(\xi_1)$, $G(\zeta_2) = G(\xi_2)$. If $R(s, a, s) + \gamma G(\xi_1) = G(\xi_1)$ then $R(s, a, s) = (1 - \gamma)G(\xi_1)$, and similarly, if $R(s, a, s) + \gamma G(\xi_2) = G(\xi_2)$ then $R(s, a, s) = (1 - \gamma)G(\xi_2)$. This is a contradiction, since $G(\xi_1) \neq G(\xi_2)$ and $\gamma \neq 1$.

Case 2: $G(\zeta_1) = G(\zeta_2) = G(\xi_1)$. If $R(s, a, s) + \gamma G(\xi_1) = G(\xi_1)$ then $R(s, a, s) = (1 - \gamma)G(\xi_1)$. Using $R(s, a, s) + \gamma G(\xi_2) = G(\xi_1)$, we get $(1 - \gamma)G(\xi_1) + \gamma G(\xi_2) = \gamma G(\xi_1)$. By rearranging, we get $\gamma(G(\xi_1) - G(\xi_2)) = 0$. This is a contradiction, since $G(\xi_1) \neq G(\xi_2)$ and $\gamma \neq 0$.

Case 3: $G(\zeta_1) = G(\zeta_2) = G(\xi_2)$. This is analogous to Case 2.

Case 4: $G(\zeta_1) = G(\xi_2)$, $G(\zeta_2) = G(\xi_1)$. If $R(s, a, s) + \gamma G(\xi_1) = G(\xi_2)$ then $R(s, a, s) = G(\xi_2) - \gamma G(\xi_2)$, and similarly, if $R(s, a, s) + \gamma G(\xi_2) = G(\xi_1)$ then $R(s, a, s) = G(\xi_1) - \gamma G(\xi_2)$. Combining this, and rearranging, gives $(1 + \gamma)G(\xi_1) = (1 + \gamma)G(\xi_2)$. This is a contradiction, since $G(\xi_1) \neq G(\xi_2)$ and $\gamma \neq -1$.

This exhausts all cases, which means that if R is non-constant, then there must be some state s and some trajectories $\zeta_1, \zeta_2, \zeta_3$ starting in s such that $G(\zeta_1) \neq G(\zeta_2)$, $G(\zeta_2) \neq G(\zeta_3)$, and $G(\zeta_1) \neq G(\zeta_3)$. Finally, note that this means that we can construct such trajectories for *any* state s', by simply composing a transition $\langle s', a, s \rangle$ with each of $\zeta_1, \zeta_2, \zeta_3$. □

Lemma 2. *If $G_2(\xi) = f(G_1(\xi))$ for all ξ and some f, then for any transition $\langle s, a, s' \rangle$ and any trajectory ζ starting in s', $R_2(s, a, s') = f(R_1(s, a, s') + \gamma G_1(\zeta)) - \gamma f(G_1(\zeta))$.*

Proof. Suppose that $G_2(\xi) = f(G_1(\xi))$ for all trajectories ξ. Let $\langle s, a, s' \rangle$ be an arbitrary transition, let ζ be an arbitrary trajectory starting in s', and let $\xi = \langle s, a, s' \rangle + \zeta$. We have that $G_2(\xi) = R_2(s, a, s') + \gamma G_2(\zeta)$, and also that $G_2(\xi) = f(G_1(\xi))$, which implies that

$$R_2(s, a, s') + \gamma G_2(\zeta) = f(G_1(\xi)).$$

Since $G_1(\xi) = R_1(s, a, s') + \gamma G_1(\zeta)$, this implies that

$$R_2(s, a, s') + \gamma G_2(\zeta) = f(R_1(s, a, s') + \gamma G_1(\zeta)).$$

By using the fact that $G_2(\zeta) = f(G_1(\zeta))$, and rearranging, we get that

$$R_2(s, a, s') = f(R_1(s, a, s') + \gamma G_1(\zeta)) - \gamma f(G_1(\zeta)).$$

Since $\langle s, a, s' \rangle$ and ζ were chosen arbitrarily, this completes the proof. □

Lemma 3. *For any non-constant reward R_1 and any f that is injective on $\mathrm{range}(G_1)$, if for any $y \in \mathrm{range}(R_1)$ and any $\gamma \in (0, 1)$ there are at most two distinct x_1, x_2 such that $f(y + \gamma x_1) - \gamma f(x_1) = f(y + \gamma x_2) - \gamma f(x_2)$ then there is no reward R_2 such that $G_2(\xi) = f(G_1(\xi))$ for all ξ.*

Proof. Suppose for contradiction that $G_2(\xi) = f(G_1(\xi))$ for all ξ. Let $\langle s, a, s' \rangle$ be an arbitrary transition. Applying Lemma 2, we get that

$$R_2(s, a, s') = f(R_1(s, a, s') + \gamma G_1(\zeta)) - \gamma f(G_1(\zeta))$$

for all trajectories ζ starting in s'. For clarity, let $x = G_1(\zeta)$ and $y = R_1(s, a, s')$, so that $f(y + \gamma x) - \gamma f(x)$. By assumption, there can be at most two distinct values x_1, x_2 such that $f(y+\gamma x_1) - \gamma f(x_1) = f(y + \gamma x_2) - \gamma f(x_2)$. However, Lemma 1 implies that there are at least three $\zeta_1, \zeta_2, \zeta_3$ starting in s' with distinct values of G_1. Since f is injective on $\text{range}(G_1)$, this means that there are at least three distinct values of x for which $f(y + \gamma x) - \gamma f(x)$ must be constant (and equal to $R_2(s, a, s')$), which is a contradiction. □

B Towards Necessary and Sufficient Conditions

In this paper, we have provided several examples of "natural" policy orderings which cannot be represented using a reward function. It would be desirable to have a set of necessary and sufficient conditions to characterise those orderings over Π that *can* be expressed by reward functions, similar to that provided by the VNM axioms (the VNM axioms themselves do not provide this, see Appendix C). We consider this to be an important topic for future work. In this section, we will discuss a few interesting properties which are shared by all policy orderings which can be represented by reward functions. We believe that these examples will help with building an intuition for what reward functions can and cannot express.

We would first like to point out that, while it seems difficult to characterise the *policy orderings* which can be expressed by reward functions, it is fairly straightforward to exactly characterise the sets of policies $\hat{\Pi}$ that can be *optimal* under some reward function:

Proposition 1. *A set of policies $\hat{\Pi}$ is the optimal policy set for some reward function 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 $\text{supp}(\pi(s)) \subseteq o(s)$.*

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. □

This immediately lets us rule out many policy orderings as inexpressible. 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 reward function under which "play rock, paper, and scissors with equal probability" is the unique optimal policy.

The next thing we would like to point out is that no reward function can express an ordering over Π that has a *countable* number of equivalence classes (except trivial reward functions, which have only one equivalence class). This simple fact also rules out many orderings.

Proposition 2. *If R is non-trivial then J has an uncountable number of equivalence classes.*

Proof. This follows from the intermediate value theorem, and the fact that J is continuous in Π. □

This simple observation can be used to e.g. create an alternative proof of Theorem 4, which says that the MaxSat objective cannot be represented as a (scalar) reward function. It also shows that objectives such as e.g. $J(\pi) = \min_{\xi \in \text{supp}(\pi)} G(\xi)$, which evaluates policies according to the worst trajectory in their support, cannot be represented (since any policy then has the same value as some deterministic policy, and since there is only a finite number of deterministic policies).

C A Digression on the Von Neumann–Morgenstern Axioms

The famous VNM axioms, due to von Neumann & Morgenstern (1947), provide necessary and sufficient conditions for when a utility function can be used to represent a preference ordering for lotteries over a finite choice set. In an MDP, a policy induces a distribution over trajectories, and a reward function assigns a value to each trajectory. One might then wonder if the VNM axioms could provide necessary and sufficient conditions for when an ordering over Π can be realised using

a reward function. This is not the case, and in this appendix, we briefly point out why. These results are not novel to this paper, but are instead provided to help with intuition building.

First of all, the VNM theorem assumes that the choice set is finite, whereas in an MDP, the number of trajectories is (countably) infinite. There are preferences between distributions over countable choice sets which satisfy the VNM axioms, but which can nonetheless not be represented using utility functions.[1] Second, not all distributions over trajectories can be represented as a policy (unless we allow both the policy and the transition function to be non-stationary). Third, there is a special structure to how a reward function assigns value to a trajectory, and not all functions $\Xi \to \mathbb{R}$ can be represented in this way. This means that the VNM axioms are not applicable to RL. However, it may still be possible to provide similar intuitive necessary and sufficient conditions for the RL case. We consider this to be an important topic for future work.

D 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 9. *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 5, 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 10. *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 11. *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 12. *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 Theorem 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 $\max_\pi J_1(\pi) \dots \max_\pi J_k(\pi)$.

[1] For example, consider the ordering that prefers all distributions with infinite support over all distributions with finite support, and which is indifferent between any two distributions in either of these classes.

E 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 13. *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.[2] 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

[2] 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.

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 algorithm 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 13, 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.