COMPARING NON-MARKOVIAN TASKS REPRESENTED BY REWARD MACHINES

Anonymous authors
Paper under double-blind review

ABSTRACT

In reinforcement learning, when reward functions are learned from data, it is useful to be able to make quantitative comparisons of reward functions. The overwhelming majority of work on metrics for comparing reward functions is focused on cases where the rewards are Markovian. On the other hand, temporally extended tasks whose rewards depend on history, where the agent has to accomplish multiple steps in a correct order, are non-Markovian. A popular way of formalising non-Markovian reward functions is by using reward machines, a type of deterministic finite automaton. Building on existing work for Markovian reward functions, we introduce a family of environment-dependent pseudometrics for comparing reward machines. We compare rewards over a common space of feasible histories, making the comparisons consistent across different representations of the same task. We show zero distance coincides with task equivalence, and we establish policy regret bounds for reward machines of bounded size. Our experiments demonstrate invariance of our method to reward machine representations. Our method provides a basis for evaluating learned reward machines by how well they represent an intended task, rather than how well they reconstruct a particular automaton.

1 INTRODUCTION

In reinforcement learning (RL), sequential decision-making problems are usually formalised using reward functions. When rewards are learned from data, it is useful to be able to quantify how different two reward functions are. There is an existing body of work on comparing Markovian reward functions, where rewards depend only on the current transition (Gleave et al., 2021; Wulfe et al., 2022). Skalse et al. (2024) introduced STARC, a class of pseudometrics for comparing Markovian reward functions which ignore differences that do not affect downstream performance.

We want to extend the idea of quantifiable comparisons to non-Markovian reward functions. Non-Markovian reward functions formalise tasks that depend on history. For example, delivering an item is only rewarded if the item was collected first. A popular way of formalising such tasks is reward machines (RMs), deterministic finite automata that use high-level events, marked by labels, emitted by the environment to keep track of relevant history (Toro Icarte et al., 2018; 2022).

One way to tell two reward machines are different is by finding a witness—a trajectory under which produced rewards are different. Two machines can differ on some sequence of labels, but this only matters if the given environment can generate such a sequence. Visible in our example shown in Figure 1, reward machines can produce identical rewards on every feasible trajectory, but they do not have to be identical. At the same time, exact equivalence is a very coarse concept. If we are training a policy under an approximated reward machine, we may be interested in how much performance we lose when deploying the policy under some other reward machine. Our goal is a fine-grained, environment-dependent notion of task difference that can provide quantifiable comparisons relevant for downstream decision making.

In this work, we study when reward machines represent the same task in a given environment and how differences between such tasks can be quantified consistently. Composing a reward machine with a Markov decision process (MDP) yields a product process on which the reward becomes Markovian, allowing us to apply existing technical machinery. We identify two different reward machines with the same task when they induce the same ordering of policies.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

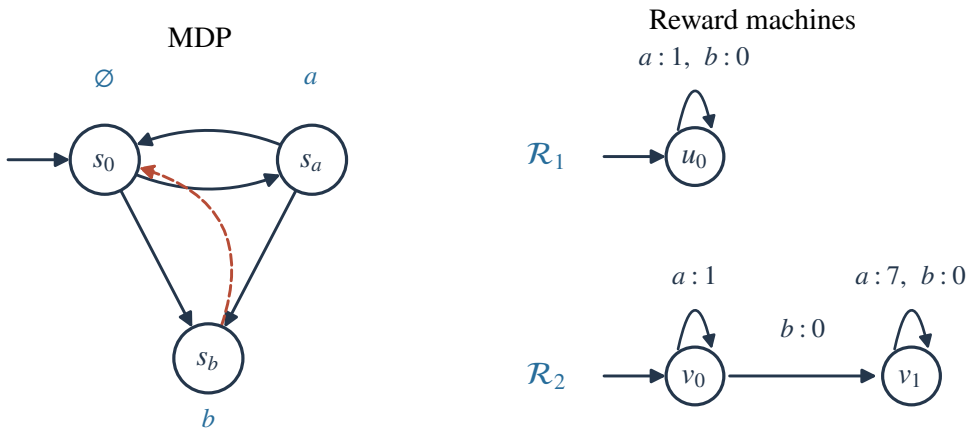


Figure 1: **Comparison depends on the environment.** On the left, we have a simple Markov decision process (MDP) with three states. With initial state s_0, moving to s_a emits label a, and moving to s_b emits label b. On the right, we have two simple reward machines. RMs $\mathcal{R}_1$ and $\mathcal{R}_2$ can only be distinguished if the environment can emit label a after emitting label b. Such a sequence is only attainable if the MDP transition marked with a dashed red arrow is possible.

Building on this view, we introduce STARM, a pseudometric for comparing reward machines. Rather than comparing automaton representations directly, STARM defines the comparison over the common space of feasible environment histories, which makes the resulting distance consistent across different representations of the same task. For deterministic environments, we also provide conditions under which task equivalence is enough to recover a minimal reward machine, connecting our work to relevant automata learning literature. Our experiments test these theoretical properties on controlled examples and benchmark environments and compare STARM with simpler alternatives.

More details on related work can be found in Appendix A.

2 Preliminaries

Setting. We consider a finite labeled Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mu_0, \gamma, \mathcal{P}, L)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P\colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the transition kernel, μ_0 the initial distribution, $\gamma \in (0,1)$ the discount factor, $\mathcal{P}$ a finite set of atomic propositions, and $L\colon \mathcal{S} \to 2^{\mathcal{P}}$ the labeling function. Here $\Delta(\mathcal{S})$ denotes the set of probability distributions over $\mathcal{S}$, so that $P(\cdot \mid s, a)$ is the distribution over successor states after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. Interaction proceeds in discrete time steps indexed by $t \geq 0$. In state $s_t \in \mathcal{S}$ the agent takes action $a_t \in \mathcal{A}$ and the environment draws a successor $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, which is labeled by $\ell_{t+1} = L(s_{t+1})$. We call the tuple $(s_t, a_t, s_{t+1}, \ell_{t+1})$ a transition. The label $L(s) \subseteq \mathcal{P}$ describes high level events in the environment which might be relevant to successful completion of the task.

The defining characteristic of non-Markovian tasks is their dependence on history. A history $h_t = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$ is a finite state-action sequence. Let Π denote the history-dependent policies $\pi(a \mid h_t)$, and let $\Pi_{\mathrm{M}} \subseteq \Pi$ denote the stationary Markovian policies $\pi(a \mid s)$. For $\pi \in \Pi$, we write $\mathbb{E}_{\mu_0, P, \pi}$ for expectation under the trajectory distribution induced by $s_0 \sim \mu_0$, $a_t \sim \pi(\cdot \mid h_t)$, and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The *label word* of a history h_t is the sequence $L(s_1), \ldots, L(s_t)$. A word is attainable if it is the label word of a history which occurs with positive probability under some policy. We write $\mathcal{P}_{\mathrm{real}}$ for the labels carried by MDP states and $\mathcal{L}$ for the attainable label language.

Reward machines. A reward machine (Toro Icarte et al., 2018) specifies a temporally extended task over the propositions $\mathcal{P}$. A reward machine is a tuple $\mathcal{R} = (U, u_0, \delta, R)$ with finite automaton state set U, initial automaton state u_0, transition function $\delta\colon U \times 2^{\mathcal{P}} \to U$, and reward function $R\colon U \times 2^{\mathcal{P}} \to \mathbb{R}$. Given a label word $\ell_1, \ell_2, \ldots, \ell_t$, the reward machine induces the state sequence

$(u_0, u_1, \ldots, u_t)$, where $u_t = \delta(u_{t-1}, \ell_t)$. At step t, the transition from u_{t-1} to u_t emits reward $R(u_{t-1}, \ell_t)$. Composed with the labeled MDP, $\mathcal{R}$ receives the labels of the labeled MDP, defining a product process. At each environment step $t \geq 1$, the label $\ell_t = L(s_t)$ of the entered MDP state is emitted and passed to the automaton. The automaton transition emits reward $R(u_{t-1}, L(s_t))$. This reward is then discounted at its arrival time by γ^{t-1}. The corresponding expected one-step reward is $\overline{R}(s, u, a) = \sum_{s' \in \mathcal{S}} P(s' \mid s, a) R(u, L(s'))$. For a history-dependent policy $\pi \in \Pi$, its return is $J_{\mathcal{R}}(\pi) = \mathbb{E}_{\mu_0, P, \pi}\left[\sum_{t \geq 1} \gamma^{t-1} R(u_{t-1}, L(s_t))\right]$, where $s_0 \sim \mu_0$ and the automaton starts at its designated initial state u_0. When comparing $\mathcal{R}_1$ and $\mathcal{R}_2$, we abbreviate $J_i = J_{\mathcal{R}_i}$. The reward depends on the automaton state, so it can be non-Markovian in s. However, it is Markovian on the augmented state (s_t, u_t). For a stationary policy $\pi(a \mid s, u)$ on the product space, the state-value function $V^{\pi}_{\mathcal{R}}(s, u)$ is defined by the same expected discounted sum as $J_{\mathcal{R}}(\pi)$, but with the product process initialized at (s, u).

Task equivalence and regret. Reward functions and reward machines are tools used to train policies to solve downstream tasks. A task is characterized by the preferences it induces over policies. We regard reward machines which induce the same ordering over all history-dependent policies as representing the same task. We say that two reward machines define the same task, denoted $\mathcal{R}_1 \sim \mathcal{R}_2$, if they induce the same ordering of all history-dependent policies:

$$J_1(\pi) \leq J_1(\pi') \quad \Longleftrightarrow \quad J_2(\pi) \leq J_2(\pi') \qquad \text{for all } \pi, \pi' \in \Pi.$$

For $\Gamma \in \{\Pi_{\mathrm{M}}, \Pi\}$, let

$$\operatorname{rng}_{\Gamma} J_i = \max_{\pi \in \Gamma} J_i(\pi) - \min_{\pi \in \Gamma} J_i(\pi).$$

The normalized worst-case regret

$$\operatorname{Reg}_{\Gamma}(J_1, J_2) = \max_{(i,j) \in \{(1,2),(2,1)\}} \sup_{\substack{\pi_a, \pi_b \in \Gamma \\ J_j(\pi_b) \geq J_j(\pi_a)}} \frac{J_i(\pi_a) - J_i(\pi_b)}{\operatorname{rng}_{\Gamma} J_i}$$

measures the largest relative loss under one reward after a policy change that does not decrease the value under the other reward. A term with zero value range is defined to be zero. For reward machines, we write $\operatorname{Reg} = \operatorname{Reg}_{\Pi}$ and $\operatorname{rng} J_i = \operatorname{rng}_{\Pi} J_i$.

Potential shaping. Reward functions can differ numerically while inducing the same policy ordering. Since we want to compare the tasks represented by rewards, and not their particular representations, we need to account for such transformations. Ng et al. (1999) introduced *potential shaping* as a reward transformation that preserves optimal policies. For a Markovian transition reward $r : S \times A \times S \to \mathbb{R}$, define $J_r(\pi) = \mathbb{E}_{\mu_0, P, \pi}\left[\sum_{t \geq 0} \gamma^t r(S_t, A_t, S_{t+1})\right]$, and let $V^{\pi}_r(s) = \mathbb{E}_{P, \pi}\left[\sum_{t \geq 0} \gamma^t r(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = s\right]$. We also write $\bar{r}(s, a) = \mathbb{E}_{S' \sim P(\cdot \mid s, a)}[r(s, a, S')]$. For a state potential $\Phi \colon \mathcal{S} \to \mathbb{R}$, potential-based reward shaping transforms a Markovian reward into $r^{\Phi}(s, a, s') = r(s, a, s') + \gamma \Phi(s') - \Phi(s)$. The discounted shaping terms telescope, giving $J_{r^{\Phi}}(\pi) = J_r(\pi) - \mathbb{E}_{S_0 \sim \mu_0}[\Phi(S_0)]$. We see that shaping preserves policy ordering.

The notion can be extended to settings with reward machines by defining potentials on RM states (Camacho et al., 2019). Given a potential function over RM states $\phi \colon U \to \mathbb{R}$, define

$$R^{\phi}(u, \ell) = R(u, \ell) + \gamma \phi(\delta(u, \ell)) - \phi(u).$$

Writing $\mathcal{R}^{\phi} = (U, u_0, \delta, R^{\phi})$ for the shaped RM, we have $J_{\mathcal{R}^{\phi}}(\pi) = J_{\mathcal{R}}(\pi) - \phi(u_0)$, and $\mathcal{R}^{\phi} \sim \mathcal{R}$. In particular, for $\delta(u, \emptyset) = u$, $R^{\phi}(u, \emptyset) = R(u, \emptyset) + (\gamma - 1)\phi(u)$, so shaping also modifies rewards on self-loops coming from empty labels.

Successor-state redistribution and positive scaling. There are some other possible mechanisms, similar to potential shaping, which modify the reward values without any downstream effect. Policy ordering is insensitive to reward function distribution over non-deterministic successor states, as long as the expectation is unchanged. Positive reward scaling also leaves policy ordering untouched. For fixed dynamics P, two Markovian rewards differ by *S'-redistribution* (Skalse et al., 2023) if $\mathbb{E}_{S' \sim P(\cdot \mid s, a)}[r_1(s, a, S')] = \mathbb{E}_{S' \sim P(\cdot \mid s, a)}[r_2(s, a, S')]$ for every (s,a). There is no change in expectation, but individual rewards assigned to transitions can be different. Using $\overline{r}$ identifies rewards that differ only by S'-redistribution. It is immediately clear none of this changes policy ordering.

For Markovian rewards on a finite MDP with all states in the state space being reachable, these transformations characterize policy-order equivalence: two rewards induce the same ordering of Π_{M} if and only if they differ only by a combination of potential shaping, positive scaling, and S'-redistribution (Skalse & Abate, 2023).

STARC pseudometrics. STARC is a class of pseudometrics for quantifying differences between Markovian reward functions. A *pseudometric* on a set X is a nonnegative, symmetric function $d\colon X \times X \to \mathbb{R}$ satisfying $d(x, x) = 0$ and the triangle inequality. A metric additionally satisfies that $d(x, y) = 0$ implies $x = y$.

STARC compares Markovian rewards by canonicalizing, normalizing, and measuring distances. Let $\mathcal{F}$ be the vector space of Markovian transition rewards on the fixed reachable MDP. Reward function $r\colon S \times A \times S \to \mathbb{R}$ can be identified with the vector of its scalar rewards on each transition, with pointwise addition and multiplication. On such a domain of reward functions, we define *canonicalization functions*. A function $c\colon \mathcal{F} \to \mathcal{F}$ is called a canonicalization function if it is linear, and it maps each reward to one differing only by potential shaping and S'-redistribution, and satisfies $c(r_1) = c(r_2)$ exactly when r_1 and r_2 differ only by those transformations.

With a norm n on $\mathrm{im}(c)$, the standardized reward is

$$s(r) = \begin{cases} c(r)/n(c(r)), & n(c(r)) > 0, \\ 0, & n(c(r)) = 0. \end{cases}$$

A metric m is *admissible* on these standardized rewards if, for a norm p and constants $0 < l \leq u < \infty$,

$$l \cdot p(x - y) \leq m(x, y) \leq u \cdot p(x - y) \qquad \text{for all } x, y \in \mathrm{im}(s).$$

A reward function distance then has the form $d_{\mathrm{STARC}}(r_1, r_2) = m\big(s(r_1), s(r_2)\big)$. After getting the standardized representation $s(r)$, distance d_{STARC} provides a scalar notion of reward difference. An inadmissible metric would be able to arbitrarily distort differences between standardized rewards. Admissibility makes sure that we have the geometric separation from which the regret guarantees follow. Canonicalization removes potential shaping and S'-redistribution, while normalization removes positive scaling. Any norm-induced distance is admissible. Note that the normalization norm and the comparison metric do not have to be the same. The norms p and n are not directly related. While n determines the normalization, the norm p is only there to witness that the metric m behaves like a norm up to constant factors.

Value-Adjusted Levelling. One possible canonicalization is *Value-Adjusted Levelling* (VAL). For a fixed reference policy $\pi_{\mathrm{ref}} \in \Pi_{\mathrm{M}}$, VAL is defined by

$$c_{\pi_{\mathrm{ref}}}(r)(s, a, s') = \bar{r}(s, a) - V_r^{\pi_{\mathrm{ref}}}(s) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a)}[V_r^{\pi_{\mathrm{ref}}}(S')].$$

For each fixed reference policy π_{ref} we have that $c_{\pi_{\mathrm{ref}}}$ is a valid canonicalization function. We fix π_{ref} throughout and write $c = c_{\pi_{\mathrm{ref}}}$. The right-hand side does not depend on the argument s', so we also write it as $c(r)(s, a)$. It is the advantage of action a under π_{ref}.

3 STARM: Comparing Reward Machines

Our goal is comparing reward machines by the tasks they represent in a given environment. We do this by abstracting away the irrelevant differences in the automaton representation. We begin by characterizing when two reward machines induce the same task by using their product with the environment, going from a non-Markovian to a Markovian problem.

3.1 Environmental task equivalence

We run two reward machines in parallel with the environment. Their synchronized product has states $x = (s, u_1, u_2)$ and transitions to $x' = (s', u_1', u_2') = (s', \delta_1(u_1, L(s')), \delta_2(u_2, L(s')))$ with probability $P(s' \mid s, a)$. The reachable product X consists of states reachable from $\mathrm{supp}(\mu_0) \times \{(u_{0,1}, u_{0,2})\}$, with initial distribution $\mu_X(s, u_1, u_2) = \mu_0(s)\,\mathbf{1}\{u_1 = u_{0,1},\ u_2 = u_{0,2}\}$. Each machine induces $\tilde{R}_i(x, a, x') = R_i(u_i, L(s'))$, where R_i corresponds to the RM whose state is denoted u_i, for $i \in \{1, 2\}$. We work with expected product rewards $\overline{R}_i(x, a) = \mathbb{E}[\tilde{R}_i(x, a, X') \mid x, a]$.

For a feasible history h_t, write $x(h_t) = (s_t, u_1(h_t), u_2(h_t))$, where $u_i(h_t)$ is the state reached by $\mathcal{R}_i$ from its initial state after reading $L(s_1), \ldots, L(s_t)$. Every stationary stochastic product policy π_m can trivially be implemented as a history-dependent environment policy by setting $\pi(a \mid h_t) = \pi_m(a \mid x(h_t))$ on feasible histories. Starting from μ_X and μ_0, respectively, the product policy and its implementation have the same expected discounted return under each of the two rewards.

Lemma 1. *$\mathcal{R}_1 \sim \mathcal{R}_2$ if and only if there are $c > 0$ and $\Phi : X \to \mathbb{R}$ such that, on every reachable (x, a),*

$$\overline{R}_2(x, a) = c\overline{R}_1(x, a) + \gamma \mathbb{E}[\Phi(X') \mid x, a] - \Phi(x).$$

The potential Φ may depend jointly on the MDP state and RM states. The proof, along with all the other proofs and more theory, is in Appendix B.

3.2 Quantitative comparison

Different reward machines can represent the same environment history using different automaton states, and the synchronized product state space depends on the pair being compared. Environment histories, however, provide a common domain. We define the distance on histories and then compute it on the finite reachable product.

Let $\mathcal{H}_t$ be the set of histories $h_t = (s_0, a_0, \ldots, a_{t-1}, s_t)$ that have positive probability from μ_0 under some policy, and let $\mathcal{H} = \bigcup_{t \geq 0} \mathcal{H}_t$. Fix a stationary reference policy $\pi_{\text{ref}}(a \mid s) > 0$ for every $s \in S$ and $a \in A$, such as the uniform policy. Give each feasible history and next action mass

$$\nu(h_t, a) = (1 - \gamma)\gamma^t \Pr_{\mu_0, P, \pi_{\text{ref}}}(h_t)\pi_{\text{ref}}(a \mid s_t).$$

These masses define a probability measure on $\mathcal{H} \times A$. They weight decisions by how often the reference policy encounters them and discount later decisions, independently of the reward machine. For a machine $\mathcal{R}$, advantage $A_{\mathcal{R}}(h_t, a)$ is the change in expected return, conditional on h_t and discounted from time t, from taking a and then following π_{ref} rather than simply following π_{ref}. Because the rewards are bounded and discount $\gamma < 1$, we have $A_{\mathcal{R}} \in L_2(\nu)$, the space of square-integrable functions. Now to remove dependence on positive reward scaling we define $z_{\mathcal{R}} = A_{\mathcal{R}} / \|A_{\mathcal{R}}\|_{L_2(\nu)}$ when the norm is nonzero, and $z_{\mathcal{R}} = 0$ otherwise. *We define the distance between two reward machines as the difference in normalized advantages*:

$$\begin{aligned} d(\mathcal{R}_1, \mathcal{R}_2) &= \left(\int_{\mathcal{H} \times A} \left(z_{\mathcal{R}_1}(h, a) - z_{\mathcal{R}_2}(h, a) \right)^2 \nu(dh, da) \right)^{1/2} \\ &= \| z_{\mathcal{R}_1} - z_{\mathcal{R}_2} \|_{L_2(\nu)}. \end{aligned} \tag{1}$$

In other words, d^2 is the expected squared difference between normalized advantages at a history-action pair sampled from ν. Since S and A are finite, $\mathcal{H}$ is countable, so the integral is a generally infinite weighted sum. For the fixed environment and reference policy, $z_{\mathcal{R}}$ depends only on $\mathcal{R}$, not on the machine against which it is compared. We call the construction in Equation 1 STARM (STARC for Reward Machines), and refer to d as the STARM distance. The effects of modifying the reference policy are shown in the right panel of Figure 2.

From histories to the finite product. Evaluating the history-space integral term by term would generally require enumerating infinitely many histories. We avoid this by using advantages, as they allow a finite representation.

On the product we use $\pi_{\text{ref}}(a \mid s, u_1, u_2) = \pi_{\text{ref}}(a \mid s)$. After fixing the next action and following policy π_{ref}, future states and rewards depend on h_t through $x(h_t)$ because of the Markov environment, deterministic RM updates, and stationary reference policy. Histories reaching the same product state have the same reference-policy advantage under each reward. We can group these histories and sum their ν-weights.

The advantage is exactly the VAL canonicalization:

$$c(\tilde{R}_i)(x, a) = \overline{R}_i(x, a) - V_i^{\pi_{\text{ref}}}(x) + \gamma \mathbb{E}[V_i^{\pi_{\text{ref}}}(X') \mid x, a]. \tag{2}$$

For every history h_t reaching x, $c(\tilde{R}_i)(x, a) = A_{\mathcal{R}_i}(h_t, a)$.

We obtain product weights by summing the masses of histories that reach each product state-action pair:

$$\omega(x,a) = \sum_{t\geq 0} \sum_{h_t:\, x(h_t)=x} \nu(h_t,a), \qquad m_{\mathrm{ref}}(x) = \sum_{a\in A} \omega(x,a).$$

Every feasible history-action pair has a positive probability, so ω is positive on every reachable product state-action pair and its weights sum to one. Since the reference policy depends only on the environment state, we have $\omega(x,a) = m_{\mathrm{ref}}(x)\pi_{\mathrm{ref}}(a \mid s)$.

Computation. For known dynamics, let P_{ref} be the row-stochastic product transition matrix under π_{ref}, and treat μ_X and m_{ref} as column vectors. Here μ_X is the initial product distribution defined in Subsection 3.1. At time t, the product-state distribution is $(P_{\mathrm{ref}}^\top)^t \mu_X$. The aggregated history weights give

$$m_{\mathrm{ref}} = (1-\gamma)\sum_{t\geq 0} \gamma^t (P_{\mathrm{ref}}^\top)^t \mu_X = (1-\gamma)(I-\gamma P_{\mathrm{ref}}^\top)^{-1}\mu_X.$$

With $r_{i,\mathrm{ref}}(x) = \sum_{a\in A} \pi_{\mathrm{ref}}(a \mid s)\overline{R}_i(x,a)$, the value vectors and reference-state weight vector are the unique solutions of

$$(I-\gamma P_{\mathrm{ref}})V_i^{\pi_{\mathrm{ref}}} = r_{i,\mathrm{ref}}, \qquad i \in \{1,2\},$$
$$(I-\gamma P_{\mathrm{ref}}^\top) m_{\mathrm{ref}} = (1-\gamma)\mu_X.$$

The systems are of size $|X| \leq |S|\,|U_1|\,|U_2|$.

Define the weighted L_2 norm $\|f\|_\omega^2 = \sum_{x\in X}\sum_{a\in A} \omega(x,a) f(x,a)^2$. Grouping histories preserves the normalization factors:

$$\|c(\tilde{R}_i)\|_\omega^2 = \int_{\mathcal{H}\times A} A_{\mathcal{R}_i}(h,a)^2\, \nu(dh,da) = \|A_{\mathcal{R}_i}\|_{L_2(\nu)}^2.$$

With $s(\tilde{R}_i) = c(\tilde{R}_i)/\|c(\tilde{R}_i)\|_\omega$ when the norm is nonzero, and $s(\tilde{R}_i) = 0$ otherwise, Equation 1 becomes

$$\begin{aligned} d(\mathcal{R}_1,\mathcal{R}_2) &= \left(\sum_{x\in X}\sum_{a\in A} \omega(x,a)\big(s(\tilde{R}_1)(x,a) - s(\tilde{R}_2)(x,a)\big)^2\right)^{1/2} \\ &= \|s(\tilde{R}_1) - s(\tilde{R}_2)\|_\omega. \end{aligned} \tag{3}$$

On each fixed product, this is a STARC distance with VAL canonicalization and the ω-weighted norm for both normalization and comparison. Expected product rewards remove successor-state redistribution, VAL removes product-state potential shaping, and normalization removes positive reward scaling.

Theorem 1. *For a fixed $\mathcal{M}$ and a full-support π_{ref}, Equation 3 defines a pseudometric on reward machines, and*

$$d(\mathcal{R}_1,\mathcal{R}_2) = 0 \iff \mathcal{R}_1 \sim \mathcal{R}_2.$$

Also, $d([\mathcal{R}_1],[\mathcal{R}_2]) := d(\mathcal{R}_1,\mathcal{R}_2)$ is a well-defined metric on environmental task-equivalence classes.

The triangle inequality follows in the common space $L_2(\nu)$, even though each pair is computed on its own product.

3.3 Consistency across products

The synchronized product used to compute the distance depends on the pair of reward machines. If we define weights separately on each product, distances become dependent on chosen representation. We avoid this by using a common history distribution. We illustrate the issue with the naive alternative of weighting all reachable product state-action pairs uniformly.

Let d_{u} denote distance using uniform weights. At $\gamma = 1/2$, start from s_0 in the MDP with states (s_0, s_a, s_b), labels $(\emptyset, a, b)$, uniform actions, and successors

	0	1	2
s_0	s_0	s_a	s_b
s_a	s_0	s_a	s_a
s_b	s_0	s_b	s_b

.

The one-state machines R_a and R_b reward a and b, respectively. A two-state copy R'_a has the same reward row as R_a in both states p, q, starts at p, transitions a to q and b to p, and self-loops on empty labels. R_a and R'_a produce identical rewards on every trajectory, but

$$d_{\mathrm{u}}(R_a, R'_a) = 0, \qquad d_{\mathrm{u}}(R_a, R_b) = \sqrt{60/19} < \sqrt{36/11} = d_{\mathrm{u}}(R'_a, R_b).$$

The product with R'_a contains two copies of s_0, so uniform product weighting changes its distance to R_b. We see that d_{u} violates the triangle inequality, and is not a pseudometric.

We can recreate the same issue without redundant RM states. Replace the two reward rows of R'_a by $(1, 0, 0)$ and $(101/100, 0, 0)$, in columns $(a, b, \emptyset)$, obtaining a minimal machine R_ϵ: both states are reachable and distinguishable by a. For a pivot-based radius search with query R_a, candidate R_ϵ, and pivot R_b, $d_{\mathrm{u}}(R_a, R_\epsilon) < 0.003107 < 0.01 < 0.031747 < |d_{\mathrm{u}}(R_a, R_b) - d_{\mathrm{u}}(R_\epsilon, R_b)|$. Applying the reverse-triangle pruning rule to d_{u} would therefore incorrectly discard R_ϵ, even though it lies within radius 0.01 of the query. On the other hand, our common-history distance satisfies $|d(R_a, R_b) - d(R_\epsilon, R_b)| < 0.000267 < d(R_a, R_\epsilon) < 0.002600$. An associated experiment appears in Subsection 4.1.

3.4 Regret guarantees

Skalse et al. (2024) show that, on a fixed finite MDP, its reward distance and normalized worst-case regret bound each other up to multiplicative constants. Their concept of *soundness* gives an upper bound on regret using distance, and *completeness* gives a corresponding lower bound. We apply these results to the reachable product, yielding analogous bounds for rewards defined by reward machines. When each reward machine has at most K states, these constants can be chosen independently of the compared reward machines. This shows the quantitative relation between our distance and normalized worst-case loss when the policy is trained under one reward machine, but evaluated on a different reward machine.

For two reward machines and their reachable product X, from the fixed initial distribution μ_X, every history-based policy $\pi \in \Pi$ has a stationary stochastic policy on the reachable part of the product X preserving expected returns under $\overline{R}_1$ and $\overline{R}_2$ (Puterman, 1994; Laroche & Tachet Des Combes, 2023). This argument is further explained in Appendix B.8. Finally, we can deduce that the two policy classes have the same normalized worst-case regret.

Proposition 1. *For every $K \in \mathbb{N}$, there exist constants $0 < L_K \leq U_K < \infty$, depending only on $\mathcal{M}$, π_{ref}, and K, such that*

$$L_K d(\mathcal{R}_1, \mathcal{R}_2) \leq \mathrm{Reg}(J_1, J_2) \leq U_K d(\mathcal{R}_1, \mathcal{R}_2)$$

holds for all reward machines $\mathcal{R}_1, \mathcal{R}_2$ with at most K states each.

The same argument gives such constants for d_{u}.

An automata learning method might not find the correct reward machine, but an approximation. If a policy is trained with the approximated RM, during deployment there can be a gap in performance compared to a policy which solves the ground truth RM. Let $\mathcal{R}_\star$ be the target machine and $\widehat{\mathcal{R}}$ a learned machine, each with at most K states, and have $J_\star$ and $\widehat{J}$ denote their respective returns. If $\pi_\star, \widehat{\pi} \in \Pi$ are the respective optimal policies for an environment, the regret definition gives

$$\frac{J_\star(\pi_\star) - J_\star(\widehat{\pi})}{\operatorname{rng} J_\star} \leq \mathrm{Reg}(J_\star, \widehat{J}) \leq U_K \, d(\mathcal{R}_\star, \widehat{\mathcal{R}}). \tag{4}$$

The upper-bound constant cannot in general be independent of machine size. In Appendix B in Proposition 2 we construct a sequence of reward-machine pairs in a fixed MDP, with a reference policy, whose distances tend to zero while their normalized worst-case regret stays one.

3.5 Task equivalence and automaton minimization

In the tabular case, trimming unreachable RM states and merging RM states with identical future rewards preserves the reward sequence on every word over $\mathcal{P}_{\mathrm{real}}$. The same goes for the values of feasible policies. We immediately see that trimming and merging does not influence the metric. The

RMs can remain different after minimization while inducing the same policy ordering. While zero distance identifies a task, it does not necessarily uniquely identify a reward machine. In Appendix C.1 we explain the sufficient conditions under which task equivalence determines the reward machine. Here, we only briefly state the main theorem.

For reward machines that self-loop on empty labels, *deshaping* replaces R by $R^\circ(u,\ell) = R(u,\ell) - \gamma\phi_R(\delta(u,\ell)) + \phi_R(u)$ with $\phi_R(u) = R(u,\emptyset)/(\gamma-1)$, which sets empty-label rewards to zero and preserves the task. Under *label coverage*, every nonempty label can be emitted from every reachable pair of RM states. Under *anchoring*, each such label has an emitting transition whose endpoints connect to a cycle of empty-label transitions.

Theorem 2. *Suppose the environment is deterministic, both reward machines self-loop on empty labels, and the two machines satisfy reachability and distinguishability after deshaping, label coverage, and anchoring. If $\mathcal{R}_1 \sim \mathcal{R}_2$, then there are a bijection $\psi : U_1 \to U_2$ and $c > 0$ such that*

$$\begin{aligned}\psi(u_{0,1}) &= u_{0,2},\\ \psi(\delta_1(u,\ell)) &= \delta_2(\psi(u),\ell),\\ R_2^\circ(\psi(u),\ell) &= cR_1^\circ(u,\ell)\end{aligned}$$

for every $u \in U_1$ and $\ell \in \mathcal{P}_{\text{real}} \cup \{\emptyset\}$.

Appendix D.4 tests Theorem 2.

4 Experiments

We test representation invariance and the relationship between STARM distance and regret.

4.1 Representation invariance

We replace the reward-emitting state in Section 3.3 with $1, 2, 4, 8, 16, 32$ copies, basically unrolling the automaton, preserving rewards. Against R_b, STARM changes by less than 1.00×10^{-15}, while d_{u} rises from 1.777 to 1.821 (Figure 2, left).

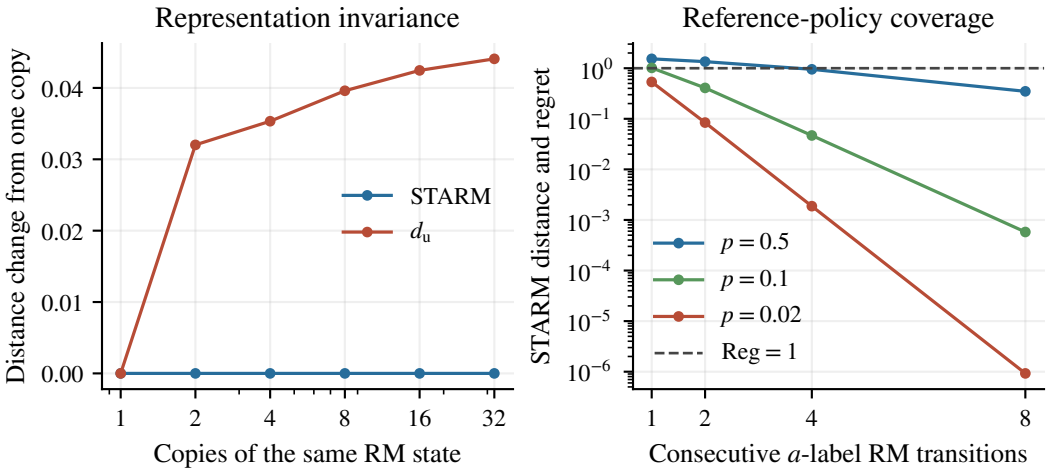


Figure 2: Left: change in distance to R_b as redundant RM states are added. We see that d_{u} changes values despite the task not changing, while STARM does not have this problem. Right: STARM distance and normalized worst-case regret for the tasks of Proposition 2, under three reference policies. We see that, while distance is influenced by the choice of reference policy, regret is invariant.

4.2 Regret sensitivity

Rare branches. The tasks of Proposition 2 can be distinguished only after k consecutive a-label RM transitions. With $\gamma = .9$, $k \in \{1, 2, 4, 8\}$ and reference probabilities $p \in \{.5, .1, .02\}$ of attaining a,

STARM decreases as k increases, while Reg stays 1 (Figure 2, right). Reducing p further decreases the distance without changing regret. Note that differences rarely encountered by the reference policy can still determine worst-case regret.

Benchmark. Using ten seeds, we evaluate 1,210 RM pairs per environment in tabular Maze (Park et al., 2025) and Office-World (Toro Icarte et al., 2018; Tasse et al., 2024). The pairs comprise five families. *Perturbation* pairs compare an RM with a reward-perturbed copy, keeping its transitions fixed. *Interpolation* pairs compare a random base RM with copies at four strengths of a shared reward perturbation. *Independent* pairs sample both RMs independently from our templates, while *acceptor* pairs restrict sampling to reach, reach-avoid, and sequence tasks. *Task-preserving* pairs apply combinations of potential shaping, positive reward scaling, state renaming, and unreachable-state additions to a base RM. More details in Appendix D.3, which also provides the complete baseline results. Table 1 compares STARM with d_{u} using Spearman correlations with Reg, computed with the linear programs outlined in Proposition 3 (Appendix B.7).

Table 1: Spearman correlation with worst-case regret Reg over all pairs. $\Delta\rho = \rho(\mathrm{STARM}) - \rho(d_{\mathrm{u}})$ is computed before rounding the displayed correlations, with 95% intervals.

	Maze			Office-World		
Family	STARM	d_{u}	$\Delta\rho$	STARM	d_{u}	$\Delta\rho$
All	0.921	0.958	$-0.036\ [-0.047, -0.028]$	0.939	0.948	$-0.010\ [-0.015, -0.004]$
Perturbation	0.905	0.927	$-0.022\ [-0.048, -0.002]$	0.891	0.886	$+0.006\ [-0.004, 0.020]$
Interpolation	0.902	0.951	$-0.049\ [-0.090, -0.017]$	0.814	0.864	$-0.050\ [-0.090, -0.014]$
Independent	0.606	0.463	$+0.144\ [0.073, 0.214]$	0.415	0.252	$+0.163\ [0.089, 0.251]$
Acceptors	0.635	0.581	$+0.054\ [-0.006, 0.108]$	0.502	0.503	$-0.001\ [-0.032, 0.021]$

STARM and d_{u} use the same VAL advantages and are different only in weighting. Pooled over families, d_{u} has the higher correlation in both environments. STARM is higher on independent-template pairs and lower on interpolations. The intervals exclude zero for both families in both environments. On perturbations and acceptors, the sign varies and three of the four intervals include zero. Two factors may contribute to this pattern. First, d_{u}'s weights depend on each pair's joint product. Independent-template pairs, where STARM wins the most, also have the widest range of reachable joint-product sizes (Table 3). Second, a worst-case policy can reach product states that the reference policy rarely visits. While d_{u} weights those states equally, STARM down-weights them, as in Proposition 2.

5 Discussion and limitations

With STARM, we extend STARC to reward machines. Our distance is invariant to different representations. We show that zero distance coincides with environmental task equivalence, obtain regret guarantees for bounded-size reward machines, and give conditions under which task equivalence recovers deshaped reward-machine structure up to isomorphism and positive scale. We empirically verify representation invariance and the structural results. We also present significant differences between simple uniform and our reference-based weighting.

A limitation of our approach is that structural recovery requires environmental coverage, along with strong assumptions like deterministic dynamics and anchoring. The relation between STARM distance and regret is not uniform as the number of states of the reward machine grows. While our experiments do support the theoretical claims, which is their goal, they do not deal with large-scale performance. Also, extending to estimated dynamics could be future work.

One motivation behind introducing STARM is its ability to help in quantifying the regret coming from training a policy under an approximated reward machine, and then deploying it under a different reward machine. Usually, algorithms that infer reward machines are based on discrete optimization, and their empirical results are focused on being correct or incorrect. Future work with STARM could focus on trying to scale reward machine inference to a very large number of states, where discrete methods become too slow and distance can be used as an objective for approximate inference.

AI USE STATEMENT

Large language models (LLMs) were used to help with phrasing in parts of the manuscript, and to check for writing errors. During earlier stages, they were used to search for related work, both for the sake of positioning the work and assessing novelty. Coding agents were used to help with implementing the experiments, and generating the figures and tables. LLMs were also used as internal reviewers and consulted when formulating and writing down the definitions and theoretical results, with a focus on rigor and consistency.

ETHICS STATEMENT

There are no societal consequences of our work that we feel must be specifically highlighted.

REPRODUCIBILITY STATEMENT

Experiments can be found in Section 4. More experiments with implementation details can be found in Appendix D. Source code is provided with the manuscript.

REFERENCES

Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 1160–1167, 1996.

Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 6065–6073, 2019.

Jan Corazza, Ivan Gavran, and Daniel Neider. Reinforcement learning with stochastic reward machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6429–6436, 2022.

Taylor Dohmen, Noah Topper, George Atia, Andre Beckus, Ashutosh Trivedi, and Alvaro Velasquez. Inferring probabilistic reward machines from non-Markovian reward signals for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pp. 574–582, 2022.

Jason Eisner. Simpler and more general minimization for weighted finite-state automata. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 64–71, 2003. URL https://aclanthology.org/N03-1009/.

Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. In *International Conference on Learning Representations*, 2021.

Erik Jenner, Joar Skalse, and Adam Gleave. A general framework for reward function distances. In *NeurIPS ML Safety Workshop*, 2022.

Romain Laroche and Remi Tachet Des Combes. On the occupancy measure of non-Markovian policies in continuous MDPs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18548–18562. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/laroche23a.html.

Michael L. Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via GLTL, 2017.

Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*. Springer, 2008. URL https://cs.nyu.edu/~mohri/pub/hbka.pdf.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606122.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

Mohamad Shehab, Antoine Aspeel, and Necmiye Ozay. Active reward machine inference from raw state trajectories, 2026.

Mohamad Louai Shehab, Antoine Aspeel, and Necmiye Ozay. Learning reward machines from partially observed policies, 2025.

Joar Skalse and Alessandro Abate. Misspecification in inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 15136–15143, 2023.

Joar Max Viktor Skalse, Matthew Farrugia-Roberts, Stuart Russell, Alessandro Abate, and Adam Gleave. Invariance in policy optimisation and partial identifiability in reward learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32033–32058. PMLR, 2023.

Joar Max Viktor Skalse, Lucy Farnik, Sumeet Ramesh Motwani, Erik Jenner, Adam Gleave, and Alessandro Abate. STARC: A general framework for quantifying differences between reward functions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wPhbtwlCDa.

Geraud Nangue Tasse, Devon Jarvis, Steven James, and Benjamin Rosman. Skill machines: Temporal logic skill composition in reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2107–2116. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/icarte18a.html.

Rodrigo Toro Icarte, Ethan Waldie, Toryn Q. Klassen, Richard Valenzano, Margarita P. Castro, and Sheila A. McIlraith. Learning reward machines for partially observable reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, January 2022. ISSN 1076-9757. doi: 10.1613/jair.1.12440. URL http://dx.doi.org/10.1613/jair.1.12440.

Blake Wulfe, Ashwin Balakrishna, Logan Ellis, Jean Mercat, Rowan McAllister, and Adrien Gaidon. Dynamics-aware comparison of learned reward functions. In *International Conference on Learning Representations*, 2022.

Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pp. 590–598, 2020.

Zhe Xu, Bo Wu, Aditya Ojha, Daniel Neider, and Ufuk Topcu. Active finite reward automaton inference and reinforcement learning using queries and counterexamples. In *Machine Learning and Knowledge Extraction (CD-MAKE)*, pp. 115–135. Springer, 2021.

A Related work

Reward comparison. Potential shaping (Ng et al., 1999), positive scaling, and successor-state reward redistribution characterize policy-order equivalence for Markovian rewards (Skalse & Abate, 2023). Skalse et al. (2023) study these and related reward transformations in a broader framework of invariance and partial identifiability in reward learning. EPIC (Gleave et al., 2021), DARD (Wulfe et al., 2022) and STARC (Skalse et al., 2024) compare Markovian reward functions. STARC also introduced worst-case regret guarantees. Jenner et al. (2022) give a related geometric treatment. In STARM we extend this line of work to reward machines by defining the comparison over feasible environment histories and computing it on synchronized products.

Reward machines. Reward machines (Toro Icarte et al., 2018; 2022) represent non-Markovian rewards using (deterministic) finite automata, building on work on non-Markovian reward specifications and temporal-logic compilation (Bacchus et al., 1996; Littman et al., 2017; Camacho et al., 2019). There are existing methods for reward machine inference (Toro Icarte et al., 2019; Xu et al., 2020; 2021), with stochastic rewards (Corazza et al., 2022; Dohmen et al., 2022), as well as from observed policies (Shehab et al., 2025; 2026). We, on the other hand, have a notion that instead talks about reward machines that induce the same ordering over feasible policies in a fixed environment.

Automata minimization. Methods for weighted automata can normalize weights before minimization (Eisner, 2003; Mohri et al., 2008). Our structural result in Appendix C is different in the sense that environmental dynamics might hide differences between abstract label sequences.

B Additional theory

B.1 Proof of Lemma 1

Proof. Suppose $\mathcal{R}_1 \sim \mathcal{R}_2$. Every stationary stochastic product policy has a history-dependent equivalent, as shown in Section 3.1. The product rewards induce the same policy ordering. The Markovian characterization (Skalse & Abate, 2023) gives positive scaling, potential shaping, and successor-state redistribution. By taking conditional expectations given (x, a), we remove the redistribution term and arrive at the stated relation for $\overline{R}_1, \overline{R}_2$.

Conversely, that relation telescopes under every $\pi \in \Pi$:

$$J_2(\pi) = cJ_1(\pi) - \mathbb{E}_{X_0 \sim \mu_X}[\Phi(X_0)].$$

The terminal potential term vanishes because Φ is bounded and $\gamma < 1$. The offset is independent of π, and $c > 0$ preserves policy ordering. □

Transformations. RM-state shaping lifts to the product potential $\Phi(x) = \phi(u_i)$. Trivially, positive scaling also preserves policy ordering. The same goes for state renaming, splitting states into transition- and reward-preserving copies, adding unreachable states, and editing entries never used on feasible histories.

B.2 Proof of Theorem 1

Proof. Fix $\mathcal{M}$ and π_{ref}. For each next action, histories reaching the same product state have the same continuation under the reference policy, so

$$A_{\mathcal{R}_i}(h_t, a) = c(\tilde{R}_i)(x(h_t), a).$$

For any product function f, we write $\hat{f}(h_t, a) = f(x(h_t), a)$. By the definition of the aggregated weights,

$$\|\hat{f}\|^2_{L_2(\nu)} = \sum_{x,a} f(x, a)^2 \sum_{t \geq 0} \sum_{h_t : x(h_t) = x} \nu(h_t, a) = \|f\|^2_{\omega}.$$

Applying this to the advantages and their linear combinations proves that Equation 1 equals Equation 3, including zero vectors. Each $z_{\mathcal{R}}$ belongs to the same $L_2(\nu)$ space, which means that nonnegativity, symmetry, and the triangle inequality follow.

Full support gives us $\omega(x, a) > 0$ on every reachable pair. We conclude that $d = 0$ exactly when $c(\tilde{R}_2) = \alpha c(\tilde{R}_1)$ for some $\alpha > 0$, including the case when both are zero. We use Equation 2 to get

$$\overline{R}_2(x, a) - \alpha \overline{R}_1(x, a) = \gamma \mathbb{E}[\Phi(X') \mid x, a] - \Phi(x),$$

where $\Phi = \alpha V_1^{\pi_{\text{ref}}} - V_2^{\pi_{\text{ref}}}$. We refer to Lemma 1 for task equivalence. In the other direction, its shaping relation implies $V_2^{\pi_{\text{ref}}} = \alpha V_1^{\pi_{\text{ref}}} - \Phi$, then $c(\tilde{R}_2) = \alpha c(\tilde{R}_1)$ and $d = 0$. We conclude that induced distance on task-equivalence classes is a metric. □

Computing the weights. Under π_{ref}, the product-state distribution at time t is $(P_{\text{ref}}^\top)^t \mu_X$. Consequently,

$$m_{\text{ref}} = (1 - \gamma) \sum_{t \geq 0} \gamma^t (P_{\text{ref}}^\top)^t \mu_X = (1 - \gamma)(I - \gamma P_{\text{ref}}^\top)^{-1} \mu_X.$$

Together with $\omega(x, a) = m_{\text{ref}}(x)\pi_{\text{ref}}(a \mid s)$, this allows us to compute the weights without having to do the possibly intractable task of evaluating individual history probabilities.

B.3 Reference policies

The advantages and history weights may use different stationary policies. The same proof applies, provided the weighting policy has full action support. The value reference does not need full action support. Using one deterministic policy for both roles would make every weighted advantage zero, since $A_{\mathcal{R}}(h, \pi(h)) = 0$. Full support prevents this from happening. Still, it does not provide us with a uniform lower bound on the weights as RM size grows.

B.4 Proof of Proposition 1

Proof. Fix $\mathcal{R}_1, \mathcal{R}_2$ and their reachable joint product X, initialized at μ_X. Every $\pi \in \Pi$ gives a history-dependent policy on X, since the RM states are fully determined by the history. Because X and A are finite and $\gamma < 1$, we know there is a stationary stochastic product policy π_m with the same discounted state-action counts (see B.8):

$$D^\pi(x, a) = \sum_{t \geq 0} \gamma^t \Pr_{\mu_X, P_X, \pi}(X_t = x, A_t = a) = D^{\pi_m}(x, a).$$

For $i \in \{1, 2\}$, recall that

$$\overline{R}_i(x, a) = \mathbb{E}[\widetilde{R}_i(x, a, X') \mid x, a].$$

By the tower property,

$$J_i(\pi) = \sum_{x,a} D^\pi(x, a) \overline{R}_i(x, a).$$

And finally,

$$J_i(\pi) = \sum_{x,a} D^{\pi_m}(x, a) \overline{R}_i(x, a) = J_i^X(\pi_m), \qquad i \in \{1, 2\}.$$

The same π_m preserves both returns because its construction depends only on the product dynamics and the original policy. Matching only the marginal discounted counts on just the MDP pair (s, a) would not be sufficient, because the policy could arrive there without following the temporally extended sequence required for a reward.

Trivially, the converse holds, that every stationary stochastic product policy has the history-dependent environment implementation described in Section 3.1. The two policy classes attain the same joint-return pairs $(J_1(\pi), J_2(\pi))$, and have the same value ranges and normalized worst-case regret. This replacement preserves total expected returns, but it does not have to preserve trajectory distributions or conditional continuation values, as it does not have to perfectly mimic behavior.

On each fixed reachable product, Equation 3 is a STARC distance with VAL canonicalization and a nondegenerate weighted L_2 norm. STARC's soundness and completeness results therefore give positive finite constants for that product, independent of the reward values.

For fixed K, only finitely many pairs of RM transition structures with at most K states occur up to state renaming. Thus only finitely many reachable product transition structures and reference-weight

systems occur. Taking the minimum of their positive lower-bound constants and the maximum of their finite upper-bound constants gives L_K and U_K. Replacing the ω-weighted norm by the unweighted one gives the same conclusion for d_{u}. □

B.5 Representation counterexample

For the construction in Section 3.3, we write $c_i = c(\tilde{R}_i)$. The unweighted inner products are

$(\mathcal{R}_1, \mathcal{R}_2)$	$\|c_1\|_2^2$	$\|c_2\|_2^2$	$\langle c_1, c_2\rangle$
(R_a, R_b)	$19/8$	$19/8$	$-11/8$
(R'_a, R_b)	$11/3$	$11/3$	$-7/3$
(R_a, R_ϵ)	$11/3$	$50275999/13500000$	$13303/3600$
(R_ϵ, R_b)	$50275999/13500000$	$11/3$	$-8459/3600$

Substitution in $d_{\mathrm{u}}^2 = 2 - 2\langle c_1, c_2\rangle/(\|c_1\|_2\|c_2\|_2)$ gives the stated square roots and

$$d_{\mathrm{u}}(R_a, R_\epsilon) \in (0.003106, 0.003107),$$
$$|d_{\mathrm{u}}(R_a, R_b) - d_{\mathrm{u}}(R_\epsilon, R_b)| \in (0.031747, 0.031748).$$

The reference-state weights are $(2/3, 1/6, 1/6)$ in the order s_0, s_a, s_b. On the four-state product, they are $(19/30, 1/30, 1/6, 1/6)$ in the order (s_0, p), (s_0, q), (s_a, q), (s_b, p). With uniform actions, the weighted inner products are

$(\mathcal{R}_1, \mathcal{R}_2)$	$\|c_1\|_\omega^2$	$\|c_2\|_\omega^2$	$\langle c_1, c_2\rangle_\omega$
(R_a, R_b)	$25/72$	$25/72$	$-17/72$
(R_a, R_ϵ)	$25/72$	$75753191/216000000$	$376877/1080000$
(R_ϵ, R_b)	$75753191/216000000$	$25/72$	$-256093/1080000$

We get $d(R_a, R_b)^2 = 84/25$, $d(R_a, R_\epsilon) \in (0.002599, 0.002600)$, and the weighted pivot difference is in $(0.000266, 0.000267)$. Exact copies have identical history advantages and therefore identical weighted comparisons.

B.6 Nonuniformity over growing automaton size

The following proposition shows that there is no uniform bound for increasing memory.

Proposition 2. *Fix $0 < p < \gamma < 1$. There is a fixed three-state MDP and a fixed full-support reference policy such that, for every $k \geq 1$, there are two $(k+1)$-state reward machines with rewards in $[0, 1]$ satisfying*

$$\mathrm{Reg}(J_{1,k}, J_{2,k}) = 1, \qquad d(\mathcal{R}_{1,k}, \mathcal{R}_{2,k}) \leq 8\sqrt{\frac{\gamma}{1-\gamma p}}\left(\frac{p}{\gamma}\right)^{k/2} \longrightarrow 0.$$

Proof. Use three environment states labeled a, b, c, with actions selecting any successor, and reference action probabilities $(p, (1-p)/2, (1-p)/2)$. Fix an initial state. Each machine starts in state zero, counts k consecutive a reads, and then enters a zero-reward absorbing state. Reading b or c enters that state immediately. Empty reads self-loop with zero reward.

Before rescaling, both machines pay one on b before termination. The second pays $2\gamma^{-(k-1)}$ on completing k consecutive a reads. Immediate b and completion have return pairs $(1, 1)$ and $(0, 2)$. Switching to completion loses the full range of the first task without decreasing the second, so $\mathrm{Reg} = 1$.

After $j < k$ consecutive a reads, the difference in reference values is $v_j = 2p^{k-j}\gamma^{-j}$. The advantage differences are $(1-p)v_j/p$ for a and $-v_j$ for b, c. This history has weight $(1-\gamma)(\gamma p)^j$, giving

$$\|A_{2,k} - A_{1,k}\|_{L_2(\nu)}^2 = \frac{4\gamma(1-\gamma)(1-p)}{1-\gamma p}\left(\frac{p}{\gamma}\right)^k [1 - (\gamma p)^k].$$

At the initial history, the first task's b, c advantages differ by one. Since $z^2 + (z-1)^2 \geq 1/2$, $\|A_{1,k}\|_{L_2(\nu)}^2 \geq (1-\gamma)(1-p)/4$. Both advantages are nonzero, and the claimed bound follows from

$$d \leq \frac{2\|A_{2,k} - A_{1,k}\|_{L_2(\nu)}}{\|A_{1,k}\|_{L_2(\nu)}}.$$

Scaling both reward tables by $\gamma^{k-1}/2$ puts their entries in $[0,1]$ without changing d or Reg. The absolute loss then equals $\gamma^{k-1}/2 \to 0$. The normalized regret remains one. □

B.7 Computing worst-case regret

Define the unnormalized discounted state-action flow polytope

$$\mathcal{D}_X = \left\{ D \geq 0 : \sum_a D(x,a) - \gamma \sum_{y,b} P_X(x \mid y,b) D(y,b) = \mu_X(x) \quad \forall x \in X \right\}.$$

Now we provide an algorithm for worst-case regret by linear programming.

Proposition 3. *The regret is the maximum of the following two linear-program values, for* $(i,j) = (1,2)$ *and* $(2,1)$*:*

$$\max_{D_a, D_b \in \mathcal{D}_X} \frac{\langle \overline{R}_i, D_a - D_b \rangle}{\operatorname{rng} J_i} \quad \textit{subject to} \quad \langle \overline{R}_j, D_b - D_a \rangle \geq 0.$$

A zero value range contributes zero.

Proof. For any policy, the discounted counts $D^\pi(x,a) = \sum_{t \geq 0} \gamma^t \Pr_{\mu_X, P_X, \pi}(X_t = x, A_t = a)$ satisfy the equations, and $J_i(\pi) = \langle \overline{R}_i, D^\pi \rangle$. Conversely, given $D \in \mathcal{D}_X$, set $\pi_D(a \mid x) = D(x,a) / \sum_b D(x,b)$ where the denominator is positive, and arbitrary values elsewhere. The state marginal of D satisfies this policy's discounted flow equation. Its solution is unique because $\gamma < 1$, so π_D generates D and has a history-dependent environment implementation. The programs optimize over the attainable joint-return pairs. The value ranges are obtained by optimizing $\overline{R}_i$ and $-\overline{R}_i$. □

Shared optimizers do not preserve policy ordering. Consider one decision with rewards $(2,1,0)$ and $(2,0,1)$ on actions a, b, c, followed by zero reward. Both tasks have the same unique optimal action a. But choosing a, b with equal probability gives returns $(3/2, 1)$, whereas choosing c gives $(0,1)$: the second task permits a switch that loses $3/4$ of the first task's range. The LP gives us $\mathrm{Reg} = 3/4$, despite the shared optimizer.

B.8 Discounted product occupancy

For a policy $\pi \in \Pi$, define its discounted product occupancy by

$$D^\pi(x,a) = \sum_{t \geq 0} \gamma^t \Pr_\pi(X_t = x, A_t = a).$$

Because the product state $X_t = (S_t, U_{1,t}, U_{2,t})$ is determined by the environment history, every history-dependent environment policy induces a history-dependent policy on the product MDP. By Puterman (1994); Laroche & Tachet Des Combes (2023), there exists a stationary stochastic product policy π_m with the same state-action occupancy:

$$D^{\pi_m} = D^\pi.$$

For either product reward $\overline{R}_i$,

$$J_i(\pi) = \sum_{x,a} D^\pi(x,a) \overline{R}_i(x,a),$$

so equality of product occupancies implies

$$J_i(\pi_m) = J_i(\pi), \qquad i \in \{1,2\}.$$

We conclude that the same stationary product policy simultaneously preserves the returns of both reward machines.

C STRUCTURAL RESULTS

Scope. Throughout this section, rewards have the table form $R(u, \ell)$ and empty labels self-loop. The comparison theory also allows rewards $r_u(s, a, s')$, transition labels $L(s, a, s')$, and memory changes on empty labels. The product, common-history argument, regret bounds, and regret linear programs apply regardless, but the structural results of this section assume tabular rewards and self-loops on empty labels.

C.1 TASK EQUIVALENCE AND AUTOMATON MINIMIZATION

We give the conditions and proof for the structural result announced in Section 3.5. Define

$$\phi_R(u) = \frac{R(u, \emptyset)}{\gamma - 1}, \qquad R^\circ(u, \ell) = R(u, \ell) - \gamma\phi_R(\delta(u, \ell)) + \phi_R(u),$$

and write $\mathcal{R}^\circ = (U, u_0, \delta, R^\circ)$ for the *deshaped* machine.

Proposition 4. *Deshaping gives $R^\circ(u, \emptyset) = 0$, $\mathcal{R}^\circ \sim \mathcal{R}$, and $(R^\phi)^\circ = R^\circ$ for every RM-state potential ϕ.*

Proof. The empty-column identity follows from $\delta(u, \emptyset) = u$. Task equivalence follows by telescoping the shaping potential $-\phi_R$. Since $\phi_{R^\phi} = \phi_R + \phi$ and shaping is additive in its potential, $(R^\phi)^\circ = R^{\phi-(\phi_R+\phi)} = R^\circ$. □

Let $W = \{(u_1, u_2) : (s, u_1, u_2) \in X \text{ for some } s\}$. A label is *realizable at* $w \in W$ if a reachable product transition from some (s, w) emits it. We use the following conditions.

Reachability and distinguishability after deshaping. Every state of each deshaped machine is reachable from its initial state by a word over $\mathcal{P}_{\text{real}}$, and every two distinct states produce different reward sequences on some word over $\mathcal{P}_{\text{real}}$.

Label coverage. Every nonempty label in $\mathcal{P}_{\text{real}}$ is realizable at every $w \in W$. The realizing environment state may depend on both w and the label.

Anchoring. For each $w \in W$, form the graph of reachable product states (s, w) with $L(s) = \emptyset$, using empty-label transitions as edges. A product state (s, w) is *anchored* if it lies in a weakly connected component of this graph containing a directed cycle, or has an action leading into such a component. For every $w \in W$ and every label realizable at w, we require a transition emitting that label whose endpoints are anchored at their respective RM-state pairs.

Proof of Theorem 2. Deshaping preserves task equivalence. By Lemma 1, there are $c > 0$ and a product potential Φ such that every reachable transition $x \to x'$ emitting ℓ satisfies

$$R_2^\circ(u_2, \ell) = cR_1^\circ(u_1, \ell) + \gamma\Phi(x') - \Phi(x).$$

Here deterministic dynamics let us write the successor directly.

Anchoring removes the potential on realizing transitions. On an empty-label edge, both rewards vanish, so $\Phi(x) = \gamma\Phi(x')$. A cycle of length k gives $\Phi(x) = \gamma^k\Phi(x)$, hence $\Phi(x) = 0$. Zero propagates in either direction along empty-label edges, and to any predecessor of the resulting component. Then Φ vanishes at all anchored states. For every realizable (w, ℓ), anchoring means we have a transition with zero potential at both endpoints, giving us

$$R_2^\circ(u_2, \ell) = cR_1^\circ(u_1, \ell), \qquad w = (u_1, u_2) \in W. \tag{5}$$

Since table rewards depend only on the RM state and label, this equality holds at every occurrence of (w, ℓ).

Label coverage. For any $w = (u_1, u_2) \in W$ and nonempty $\ell \in \mathcal{P}_{\text{real}}$, a realizing transition witnesses $(\delta_1(u_1, \ell), \delta_2(u_2, \ell)) \in W$. The empty label leaves w unchanged. This means that W is closed under every label, and equation 5 holds for $W \times (\mathcal{P}_{\text{real}} \cup \{\emptyset\})$. The transitions may belong to different environment histories, as the whole continuation does not have to be realized by a single trajectory.

Distinguishability makes the relation a bijection. Suppose $(u_1, u_2), (u_1, u_2') \in W$. Closure and equation 5 imply that u_2 and u_2' produce the same reward sequence on every continuation word. Distinguishability in the second deshaped machine gives $u_2 = u_2'$. The symmetric argument, using $c > 0$, gives uniqueness in the other coordinate. Since W contains the initial pair and is closed under labels, reachability of every RM state makes both projections of W surjective. Thus W is the graph of a bijection ψ. The initial pair, closure, and equation 5 give the three stated identities. □

These identities imply task equivalence without anchoring, label coverage, or minimality. For the original tables they give

$$R_2(\psi(u), \ell) = (cR_1)^{\phi}(u, \ell), \qquad \phi(u) = \phi_{R_2}(\psi(u)) - c\phi_{R_1}(u).$$

State renaming, positive scaling, and RM-state shaping preserve policy ordering, including stochastic environments.

Full-language coverage. Realizability of every word over $\mathcal{P}_{\mathrm{real}}$ implies label coverage: append each label to a word reaching w. The resulting history attains that label at w. The other direction does not have to hold, as the irreversible example below shows. These conditions are sufficient.

Corollary 1. *Under deterministic dynamics and anchoring, $d(\mathcal{R}_1, \mathcal{R}_2) = 0$ exactly when the expected product rewards $\overline{R}_i'$ induced by $\mathcal{R}_i^{\circ}$ are positively proportional. Consequently,*

$$d_{\mathrm{raw}}(\mathcal{R}_1, \mathcal{R}_2) = \left\| \frac{\overline{R}_1'}{\|\overline{R}_1'\|_\omega} - \frac{\overline{R}_2'}{\|\overline{R}_2'\|_\omega} \right\|_\omega$$

has the same zero set as d, with zero vectors mapped to zero. This result does not require label coverage or minimality.

Proof. For necessity, the anchoring part of the preceding proof gives equation 5 on every reachable transition, which leads to $\overline{R}_2' = c\overline{R}_1'$. Conversely, the proportionality gives the original expected rewards a product potential $\phi_{R_2}(u_2) - c\phi_{R_1}(u_1)$, so task equivalence follows from Lemma 1. Positive ω makes equality of the normalized vectors equivalent to positive proportionality, including the case when both are zero. □

Relation to VAL. On the auxiliary MDP with states U, label actions, and transitions δ, always choosing $\emptyset$ has value $V_\emptyset(u) = R(u, \emptyset)/(1-\gamma) = -\phi_R(u)$. Under this policy, VAL gives R°. Such a policy may not exist in a given MDP. VAL on the environmental product can also remove potentials depending on the environment state.

C.2 Constructive equivalence test

Corollary 2. *Fix γ and an ordering of $\mathcal{P}_{\mathrm{real}}$. Construct $C(\mathcal{R})$ on $\mathcal{P}_{\mathrm{real}} \cup \{\emptyset\}$ as follows:*

1. *Deshape the reward table.*
2. *Trim and minimize over $\mathcal{P}_{\mathrm{real}}$ by equality of output sequences.*
3. *Divide nonzero rewards by the table's maximum absolute entry, leaving zeros.*
4. *Number states by breadth-first search (BFS) from the initial state in fixed label order, and serialize the transitions and normalized rewards.*

Equal serializations imply task equivalence in any environment with this label alphabet. Under deterministic dynamics, the converse holds when anchoring and label coverage hold on the pair of minimized machines.

Proof. Let Q_i be the deshaped, trimmed, minimized machines. These operations preserve task equivalence, so equal serializations imply task equivalence of the original machines, even under stochastic dynamics. For necessity, Q_1, Q_2 satisfy reachability and distinguishability by construction. Under the two environment conditions, Theorem 2 gives an initial-state-preserving isomorphism and positively proportional rewards. Reward normalization removes the scale, and the same serialization follows from fixed-order breadth-first numbering. Special case is that a zero table minimizes to a one-state zero machine. □

C.3 Examples of the assumptions

The following examples use $\gamma = 1/2$, zero rewards and self-loops on empty-labels unless stated otherwise.

Without anchoring. Take two environment states labeled a, b, with actions selecting either successor from either state. Every word is realizable, but there are no anchors. The one-state reward rows $(1, 0, 0)$ and $(2, 1, 0)$, in columns $(a, b, \emptyset)$, are deshaped and minimal. Their actual rewards differ by one at every step, so their returns differ by the policy-independent constant 2. Although the tables are not proportional, $d = 0$. The product potential is the constant -2, as an RM-state potential cannot remove the difference while preserving the zero empty columns.

Without label coverage. Use states (s_0, s_a, s_b, t), labels $(\emptyset, a, b, \emptyset)$, initial state s_0, and deterministic successors

	0	1	2
s_0	s_0	s_a	s_b
s_a	s_0	s_a	s_b
s_b	t	s_b	s_b
t	t	s_b	s_b

.

Both irreversible phases have an empty self-loop and the anchoring assumption is true. Compare a one-state machine returning 1 on a and 0 on b with a two-state machine returning the same until its first b, then 7 on a and 0 on b. Both are deshaped and minimal. The machines agree on all feasible trajectories, since no a follows a b, but their state counts differ. The post-b RM pair cannot attain a, so label coverage is not true.

Label coverage without full-language realizability. In the same environment, start in RM state 0 and use

$$\delta = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad R = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix},$$

with transition columns (a, b) and reward columns $(a, b, \emptyset)$. The RM toggles on a, so both RM states occur before the first b. Compare it with a copy obtained by shaping with $(2, -4)$, scaling by three, and renaming states. Both reachable RM pairs have anchored transitions for a and b. Anchoring and label coverage hold although ba is not attainable. The correspondence has scale 3 and combined shaping potential $(6, -12)$. Changing the original RM's second a reward from 2 to 3 while leaving the copy fixed preserves the transition checks but gives unequal canonical forms, certifying inequivalence.

Minimality must follow deshaping. In a three-state environment labeled $(\emptyset, a, b)$, let actions select any successor. Anchoring and full-language coverage hold. Copy the one-state reward row $(1, 0, 0)$ into states p, q, start at p, and set $\delta(p, a) = \delta(q, a) = q$ and $\delta(p, b) = \delta(q, b) = p$. Shaping by $\phi(p) = 0$, $\phi(q) = 2$ gives rows $(2, 0, 0)$ and $(0, -2, -1)$. The raw machine is minimal, but deshaping restores identical outputs and minimization merges the two states (Figure 3). Simple minimality is therefore not sufficient for Theorem 2.

C.4 Raw comparison under anchoring

Let $\mathcal{S}$ be the space of expected rewards on a fixed deterministic reachable product that depend only on (w, ℓ) and vanish on empty labels. Let T be VAL restricted to $\mathcal{S}$, with $\|\cdot\|_\omega$ in both domain and codomain.

Proposition 5. *Under anchoring, T is injective. Let κ be the ratio of its largest to smallest singular value, setting $\kappa = 1$ when $\mathcal{S} = \{0\}$. Then*

$$(2\kappa)^{-1} d_{\mathrm{raw}} \leq d \leq 2\kappa d_{\mathrm{raw}}.$$

For a fixed environment and reference policy, d_{raw} also has reward-independent regret bounds on the anchored class with bounded RM size. This does not require label coverage and minimality.

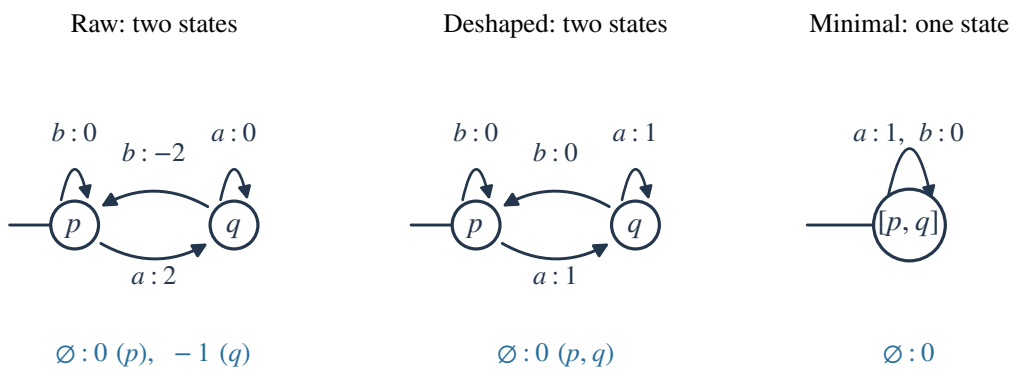


Figure 3: Deshaping before minimization. At $\gamma = 1/2$, shaping redundant states by $\phi(p) = 0$, $\phi(q) = 2$ gives distinct raw reward rows. Deshaping restores identical output behavior, allowing the states to merge. All three machines define the same task, with self-looping on empty labels.

Proof. If $Tr = 0$, then r is a product-potential difference. Since r vanishes on empty labels, its potential vanishes at anchored states by the argument in Theorem 2. Anchoring means there is a transition with anchored endpoints for every realizable (w, ℓ), so r is zero there. Since r depends only on (w, ℓ), it vanishes throughout the reachable product. We conclude that T is injective.

For nonzero $f, g \in \mathcal{S}$, write $\hat{f} = f/\|f\|_\omega$ and $\hat{g} = g/\|g\|_\omega$. The normalization inequality and the extreme singular values give

$$\left\| \frac{Tf}{\|Tf\|_\omega} - \frac{Tg}{\|Tg\|_\omega} \right\|_\omega \leq \frac{2\|T(\hat{f} - \hat{g})\|_\omega}{\|T\hat{f}\|_\omega} \leq 2\kappa \|\hat{f} - \hat{g}\|_\omega.$$

Apply the same inequality to T^{-1} on its image for the reverse bound. If exactly one input is zero, both normalized distances equal one; if both are zero, both vanish. The zero subspace is therefore covered too. Deshaping leaves VAL unchanged, so applying these bounds to $\overline{R}_1', \overline{R}_2'$ proves the claimed comparison of d and d_{raw}.

Let $L, U > 0$ be the fixed-product regret constants used in the proof of Proposition 1. Combining the bounds gives

$$\frac{L}{2\kappa} d_{\text{raw}} \leq \text{Reg} \leq 2\kappa U d_{\text{raw}}.$$

For bounded RM size, only finitely many product transition structures occur up to state renaming. Taking extrema over those satisfying anchoring gives positive finite constants independent of the rewards. □

D Experimental details

D.1 Experimental conventions

We use known dynamics and a uniform reference policy on environment states unless stated otherwise.

Numerical VAL distances use a relative zero tolerance of 10^{-9}, with the weighted or unweighted L_2 norm used by the corresponding distance. For the resulting advantage vector z, numerical zero means $\|z\|_2 \leq 10^{-12}\sqrt{\dim z}$. Nonzero vectors are normalized in Euclidean norm and a gap below 10^{-8} counts as a match. The cached STARM comparator uses this zero test before reference-weighted normalization, with the same distance threshold.

The structural raw comparison normalizes deshaped rewards and uses a 10^{-8} distance threshold. The product-potential LP uses decision tolerance 10^{-8} and feasibility tolerances 10^{-9}; solver failure or an excessive residual gives an unresolved result. Worst-case-regret LPs use HiGHS with its default solver tolerances and treat value ranges at most 10^{-9} as zero.

For rank statistics, distances and regrets are rounded to ten decimals to group numerical approximations to theoretical ties. Positive-regret pairs satisfy $\mathrm{Reg} > 10^{-8}$. Correlation differences are computed before rounding for display. We form 95% percentile intervals from 2,000 bootstrap resamples of the ten seeds within each environment. Sampling seeds with replacement keeps all pairs from each selected seed together. The same resamples are used for every comparator and ranks are recomputed.

D.2 Representation invariance and reference-policy controls

Redundant memory. In the copy example of Section 3.3, use $1, 2, 4, 8, 16, 32$ identical reward-emitting copies: a advances, stopping at the last copy, and b resets to the first. In four tasks, STARM changes by less than 1.00×10^{-15}. Against R_b, d_{u} changes from 1.777 to 1.821 (Figure 2, left panel).

Shaping versus weighting. For 12 machines per environment, we allow rewards to depend on both MDP and RM states and add MDP-state potentials. VAL comparisons remove the shaping, whereas reference-weighted raw distances range from 1.15 to 1.46 (Table 2).

Table 2: Distance ranges. The raw comparisons omit VAL and use reference (d_{raw}) or uniform ($d_{\mathrm{raw,u}}$) weights. Zero means below 10^{-10}. Shaping regret is below 10^{-12}.

Environment	Edit	d	d_{u}	d_{raw}	$d_{\mathrm{raw,u}}$
Maze	Shaping	0	0	1.15–1.46	1.19–1.40
Maze	Noise	1.04–1.46	1.18–1.45	0.82–1.40	0.92–1.31
Office-World	Shaping	0	0	1.26–1.40	1.15–1.39
Office-World	Noise	1.21–1.53	1.23–1.54	1.11–1.38	0.91–1.35

Rare branches. For Appendix B.6's construction, we use $\gamma = 0.9$, $k \in \{1, 2, 4, 8\}$ and $p \in \{.5, .1, .02\}$. At depth eight, distances are 0.349, 5.78×10^{-4} and 9.25×10^{-7}, respectively, while normalized worst-case regret remains one. We check this by using all 30 combinations of $\gamma \in \{0.5, 0.9\}$, $p \in \{.02, .1, .4\}$ and $k \in \{1, 2, 4, 8, 16\}$, canonicalizing the reward difference.

D.3 Benchmark setup and full results

Setup. Maze uses the OGBench medium-maze layout (Park et al., 2025), with each free coarse cell subdivided into a 2×2 block. It has 104 states, five actions and $\gamma = 0.98$. Office-World follows the reward-machine benchmark of Toro Icarte et al. (2018), using the static map and predicates of Skill Machines (Tasse et al., 2024). It has 120 states, four actions and $\gamma = 0.99$. Both have deterministic, continuing dynamics and zero-based grid coordinates. Maze starts are uniform over fine cells in coarse cells $(4, 1), (4, 2), (5, 1)$. Office-World starts are uniform over the twelve cells in row 1 at columns $1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15$. Ten seeds each generate 121 pairs. The templates contain reach, reach-avoid, sequence, patrol, diamond and random machines, with sampled labels and reward parameters. The five pair families are:

- *Perturbation* (40 per seed). Compare a sampled template machine with a copy whose rewards on transitions that change RM state receive independent $\mathcal{N}(0, \sigma^2)$ noise, choosing σ uniformly from $\{0.25, 0.5, 1, 2\}$.
- *Interpolation* (24 per seed). For six random machines with 2–4 states, compare each reward table R with $(1 - \lambda)R + \lambda R'$ at $\lambda \in \{0.25, 0.5, 0.75, 1\}$, where R' is one copy perturbed with $\sigma = 3$ as above. Keep the transition function fixed.
- *Independent templates* (25 per seed). Sample two template types uniformly with replacement, then independently sample a machine of each type.
- *Task-preserving* (12 per seed). Compare a sampled template machine with a transformed copy. Apply potential shaping and positive scaling independently, each with probability 0.8. Permute states with probability 0.5 and add an unreachable state with probability 0.3. Potential entries are independent $\mathcal{N}(0, 2^2)$ draws; the scale is uniform on $[0.3, 4)$.
- *Acceptors* (20 per seed). Independently sample two machines from reach, reach-avoid and sequence, selecting each template type uniformly. They have absorbing acceptance states entered

with positive reward. Sequence machines also reward intermediate progress; reach-avoid also has an absorbing rejection state entered with a penalty.

Both start sets pass the language-coverage check with empty-label steps omitted, and all sampled pairs pass anchoring.

Baselines. The baseline d_{u} uses the same VAL advantages as STARM with uniform weights. The raw distances remove RM-state potential shaping using the empty-label rewards, then compare normalized product rewards without VAL. They use reference weights for d_{raw} and uniform weights for $d_{\mathrm{raw,u}}$.

Graph edit distance (GED) minimizes normalized labeled-edge mismatch and reward-gap costs over initial-state-preserving node injections, omitting self-loops. Language distance is the Pearson distance between discounted returns on 256 uniform-policy traces of 200 steps each. The product-space EPIC extension (Gleave et al., 2021) samples successor coordinates independently on the reachable joint product. These samples can violate RM updates, preventing potential shaping from cancelling.

Joint-product sizes. For the same 2,420 seeded pairs, Table 3 reports the minimum and maximum numbers of reachable joint-product states before minimization.

Table 3: Reachable joint-product state counts across the ten benchmark seeds, reported as minimum–maximum within each family.

Family	Maze $\lvert X\rvert$	Office-World $\lvert X\rvert$
Perturbation	196–500	229–595
Interpolation	192–408	230–472
Independent	204–1656	352–2945
Acceptors	204–1384	234–1897
Task-preserving	196–500	232–595

Full results. Table 4 reports Spearman correlations with worst-case regret Reg, computed by the linear programs of Proposition 3, and optimizer regret

$$\mathrm{Reg}_{\mathrm{opt}}(J_1, J_2) = \frac{1}{2} \sum_{i \neq j} \frac{\max_{\pi \in \Pi} J_i(\pi) - J_i(\pi_j^*)}{\mathrm{rng}\, J_i}, \tag{6}$$

where π_j^* maximizes J_j. Worst-case regret is the primary target: shared optimizers need not imply the same policy ordering (Appendix B.7).

Without the ten-decimal rounding of Appendix D.1, pooled STARM correlations are 0.905 and 0.918, versus 0.921 and 0.939 after rounding. The corresponding pooled d_{u} correlations are 0.939 and 0.925.

On positive-regret pairs, STARM correlations are 0.852 and 0.899, versus 0.810 and 0.867 for reference-weighted raw, with 982 and 1027 pairs respectively. Table 5 gives the family breakdown. Task-preserving controls have constant regret, so their correlations are undefined; the other families also contain equivalent pairs.

Table 4: Pooled Spearman correlations with worst-case regret Reg and optimizer regret $\mathrm{Reg}_{\mathrm{opt}}$.

	Maze		Office-World	
Comparison	Reg	$\mathrm{Reg}_{\mathrm{opt}}$	Reg	$\mathrm{Reg}_{\mathrm{opt}}$
STARM d	0.921	0.819	0.939	0.859
d_{u}	0.958	0.852	0.948	0.860
d_{raw}	0.899	0.803	0.920	0.831
$d_{\mathrm{raw,u}}$	0.940	0.836	0.939	0.835
Graph edit distance	0.690	0.716	0.708	0.720
Language distance	0.802	0.766	0.861	0.823
EPIC extension	0.839	0.796	0.834	0.791

Table 5: Spearman correlations with worst-case regret, by family. Cells give all pairs / positive-regret pairs; N/N_+ gives their counts. GED denotes graph edit distance. Dashes indicate a constant target or empty subset. Rounding and positivity follow Appendix D.1.

Maze					
Family	N/N_+	d	d_{u}	d_{raw}	GED
All	1210/982	0.921/0.852	0.958/0.920	0.899/0.810	0.690/0.849
Perturbation	400/320	0.905/0.817	0.927/0.859	0.883/0.774	0.419/0.653
Interpolation	240/217	0.902/0.868	0.951/0.934	0.714/0.614	0.344/0.372
Independent	250/248	0.606/0.594	0.463/0.445	0.620/0.608	0.529/0.514
Acceptors	200/197	0.635/0.613	0.581/0.555	0.664/0.643	0.691/0.669
Task-preserving	120/0	–/–	–/–	–/–	–/–
Office-World					
Family	N/N_+	d	d_{u}	d_{raw}	GED
All	1210/1027	0.939/0.899	0.948/0.915	0.920/0.867	0.708/0.883
Perturbation	400/342	0.891/0.827	0.886/0.818	0.836/0.739	0.437/0.574
Interpolation	240/240	0.814/0.814	0.864/0.864	0.699/0.699	0.476/0.476
Independent	250/250	0.415/0.415	0.252/0.252	0.452/0.452	0.452/0.452
Acceptors	200/195	0.502/0.440	0.503/0.441	0.596/0.533	0.684/0.622
Task-preserving	120/0	–/–	–/–	–/–	–/–

For $d > 0.02$, observed regret/distance ratios range from 0.12 to 22.29 on Maze and 0.07 to 8.82 on Office-World. These extrema describe the sampled pairs.

Uncertainty. Using the resampling procedure in Appendix D.1, Table 1 gives all-pair intervals for $\rho(\mathrm{STARM}) - \rho(d_{\mathrm{u}})$. Table 6 gives the corresponding differences against reference-weighted raw, also over all pairs. These intervals condition on the fixed environments and template bank.

Table 6: STARM minus reference-weighted raw Spearman correlation on all pairs, with 95% seed-block bootstrap intervals. Positive differences favor STARM.

Family	Maze	Office-World
All	$0.022\ [0.013, 0.030]$	$0.019\ [0.012, 0.028]$
Perturbation	$0.022\ [0.003, 0.040]$	$0.055\ [0.039, 0.072]$
Interpolation	$0.188\ [0.133, 0.254]$	$0.115\ [0.043, 0.184]$
Independent	$-0.013\ [-0.057, 0.016]$	$-0.038\ [-0.081, 0.001]$
Acceptors	$-0.029\ [-0.054, -0.010]$	$-0.093\ [-0.133, -0.060]$

D.4 Structural decision experiment

Population. We evaluate Corollary 2 on 45 queries in each of six deterministic environments, with $\gamma = 1/2$ and initial state zero. The three-state hub, labeled $(\emptyset, a, b)$, allows every successor from every state. Corridors of lengths $h = 3, 6$ have an empty hub and two length-h branches ending in a, b. Action zero returns to the hub; the other actions select a branch at the hub and advance along it, then remain at its terminal label. These environments satisfy anchoring and have full-language coverage after erasing empty-label steps. The irreversible environment is the four-state example in Appendix C.3. The forced-prefix environment has labels $(\emptyset, a, \emptyset, a, b)$ and successor rows $(1,1,1), (2,3,4), (2,3,4), (2,3,4), (2,3,4)$, forcing an initial a read. The no-anchors environment has two states labeled (a, b) and permits either successor from either state.

The four reference machines are cycles of sizes $k = 2, 3, 4$, with $\delta(u, a) = u + 1 \pmod k$, $\delta(u, b) = u$, $R(u, a) = u + 1$, $R(u, b) = 1/4$, and a phase tracker

$$\delta = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \qquad R = \begin{pmatrix} 1 & 1/2 & 0 \\ 2 & 1/4 & 0 \end{pmatrix}.$$

The transition columns are a, b and the reward columns are $a, b, \emptyset$; empty reads self-loop and initially pay zero. Each reference receives seven edits: shaping by $2u - 2$ followed by seeded renaming;

Table 7: Decisions on 45 queries per environment: equivalent / inequivalent / inconclusive (E/I/?). Canonical certificates are exact; raw decisions use Appendix D.1. Pairs gives passing/tested minimized pairs for anchoring and label coverage. Full language means that every finite word over nonempty realizable labels is attainable after erasing empty-label steps.

Environment	Full language	Pairs	Canonical E/I/?	Raw E/I/?
Hub	yes	8/8	28/17/0	28/17/0
Corridor 3	yes	8/8	28/17/0	28/17/0
Corridor 6	yes	8/8	28/17/0	28/17/0
Irreversible	no	6/8	28/12/5	29/16/0
Forced prefix	no	7/8	28/16/1	28/16/1
No anchors	yes	0/8	28/0/17	28/0/17

scaling by two; two copies per state followed by shaping; an unreachable state; adding $1/2$ to the initial a reward; negative scaling; and an initial a-successor edit. We compare each reference with its seven variants and compare the three pairs among its shaped, scaled and shaped-copy variants. Four further queries add one to all nonempty rewards, one per reference; a final query adds $1/2$ to the phase tracker's post-b a reward. This gives 45 queries over 37 machines per environment. The final edit is hidden by irreversible dynamics, and the nonempty bonus is a constant offset in the no-anchors environment.

Procedures. Canonical equality certifies task equivalence. Unequal forms certify inequivalence when anchoring and label coverage hold on the minimized pair. The deshaped raw test needs only anchoring for an inequivalence decision. Ordinary Mealy equality implies task equivalence, but its inequality is inconclusive. Product checks use STARM and the LP

$$\max_{\alpha,\beta,\Psi,t} t \quad \text{subject to} \quad \alpha\overline{R}_1 - \beta\overline{R}_2 + H\Psi = 0, \qquad \alpha + \beta = 1, \qquad \alpha, \beta \geq t \geq 0,$$

where $(H\Phi)(x,a) = \gamma\mathbb{E}[\Phi(X') \mid x,a] - \Phi(x)$. A positive optimum gives $c = \alpha/\beta$ and $\Phi = \Psi/\beta$; conversely, a valid scale and potential give positive t after division by $1 + c$. Reward rescaling for conditioning is undone in the certificate.

Results. All 168 canonical equality certificates satisfy the original rational transition and reward identities. The procedure also certifies 79 inequalities, leaving 23 queries unresolved (Table 7). Every definitive answer agrees with the product checks, which find 173 equalities and 97 inequalities; the smallest nonzero STARM distance is 0.00707 and the smallest positive normalized LP margin is 0.1818. The raw test resolves the five irreversible cases left open by label coverage. Both partial tests leave the forced-prefix transition edit unresolved. Without anchors, the unresolved cases comprise four constant-offset equalities and thirteen inequalities. The bank's 116 states reduce to 105 after trimming and 95 after minimization. Its 19 canonical forms can still separate task-equivalent machines when the conditions fail.

D.5 Computational costs

The quotient baseline computes reference advantages on each machine's own product, aligns them on the joint product and tests positive proportionality. It caches sparse value factorizations by transition structure, advantages by input contents, and alignment graphs; identical structures need no extra joint graph. It needs no reference-weight solve. Cached STARM uses the same caches and additionally caches reference weights.

The benchmark has 199 exact canonical equalities and 411 numerical quotient matches. The 212 additional matches are task-preserving controls whose floating-point reward tables no longer satisfy the exact transformation identities.

The exact-first cascade tries canonical equality or certified structural inequality, then the anchored raw test, then numerical quotient fallback. Positive screening uses only exact equalities before fallback, preserving every quotient answer on these pairs. Exact-first can instead reject rounded near-matches. Each method uses its applicable caches, but no method caches a pair's answer. Inconclusive exact tests are followed by numerical fallback.

We time all 270 constructed queries and 2,420 benchmark pairs in six control batches and four benchmark batches, separating identical and differing transition structures. Each batch starts with empty caches and is replayed warm, over three repetitions with rotated method order and one BLAS thread. Timings include preprocessing, product construction, graph checks and fallback, but exclude input generation, independent certificate verification, imports and allocation tracing. Positive screening reduces the small-control cold total, but is slower than the quotient alone on the full benchmark, both cold and warm (Table 8). The tested cascades do not show an overall benchmark speedup.

Table 8: Complete-decision runtimes, summing median batch times over three repetitions. E/N denotes exact/numerical answers; E+ uses only exact positive certificates before fallback. C/W denotes cold/warm caches. † Exact-first can reject numerical near-matches and therefore answers a different question.

Procedure	Answers	270 C/W (ms)	2,420 C/W (s)
Exact-first cascade†	E/N	81.2/4.5	26.65/0.02
Cached quotient	N	115.0/5.3	26.16/0.10
Positive certificates + quotient	E+/N	67.5/5.0	27.01/0.11
Raw + quotient fallback	N	87.9/11.6	27.06/0.27
Cached STARM	N	175.7/9.7	28.14/0.15