Transfer Learning between non-Markovian RL Tasks through Semantic Representations of Temporal States

Andrea Fanti[1,*], Elena Umili[1] and Roberto Capobianco[2]

[1]Sapienza University of Rome
[2]Sony AI

Abstract

Reinforcement Learning (RL) faces challenges in transferring knowledge across tasks efficiently. In this work we focus on transferring policies between different temporally extended tasks expressed in Linear Temporal Logic. Existing methodologies either rely on task-specific representations or train deep-neural-networks-based task-state representations exploiting the interaction with the environment. We propose a novel approach, leveraging semantic similarity of the formulas, to compute transferable task state representations directly from task specifications, offline, and without any learning process. Preliminary experiments on temporally-extended navigation in a grid world domain demonstrate the superiority of our semantic representation over baseline methods. This approach lays the groundwork for lightweight, transferable task state representations based solely on task semantics, offering a promising avenue for efficient RL in temporally-extended tasks without extensive retraining.

Keywords

Non-Markovian Reinforcement Learning, Instruction following, Transfer learning in RL

1. Introduction

Reinforcement Learning (RL) is a powerful Artificial Intelligence paradigm that can solve a great variety of complex decision problems with minimal domain knowledge [1]. In this work, we focus on the age-old problem of using Reinforcement Learning to make an artificial agent follow *temporally-extended instructions*, such as "*Do A and then B*", without training it on every possible task. This setting poses two major challenges to standard RL algorithms: (i) it breaks the so-called *Markov property* [2], which prevents their application to temporally-extended goals; and (ii) it requires *zero-shot transfer learning* — i.e., solving unseen tasks without any additional training.

Several methodologies have been proposed to apply RL to temporally-extended tasks exploiting their specification in formal languages [3], or finite state machines [4]. Despite their success on solving single tasks, however, these approaches aggravate the transfer learning problem, as they are based on task state representations that are specific to each task, and cannot be transferred to new, unseen tasks. To overcome this limitation, more recent work focuses on exploiting the power of Deep Neural Networks to learn latent temporal state representations using the experience collected in the environment, showing promising results [5] [6]. However, we argue that the correlation between different temporally-extended tasks, when expressed formally through instructions, depends solely on the *semantics* of the instructions. This correlation can be computed *offline* through semantic analysis, independently of the environment and agent-environment interaction. Therefore, in this work, we explore a radically different approach, in which a transferable task state representation is not learned during model training, and is instead directly computed from the task specification as a vector of real numbers. This computation is based on evaluating the semantic similarity between the task state and the initial state of one or more reference tasks, which form the components of the resulting vector space. Inspired by [7] and [8], we use a kernel function to compute the semantic similarity, and thus refer to the final result as a *Kernel Representation*.

AAPEI '24: 1st International Workshop on Adjustable Autonomy and Physical Embodied Intelligence, October 20, 2024, Santiago de Compostela, Spain.
*Corresponding author.
fanti@diag.uniroma1.it (A. Fanti); umili@diag.uniroma1.it (E. Umili); Roberto.Capobianco@sony.com (R. Capobianco)

An immediate benefit of this approach is eliminating the need to employ ad-hoc model architectures, such as Graph Neural Networks in [5]. On the other hand, an immediate limitation is that computing the exact value of a Kernel Representation on all possible traces is intractable, meaning it must be approximated. To tackle this challenge, we begin by proposing two metrics to asses the quality of an approximated representation without collecting any experience from the environment. Then, we introduce a method to reduce the number of traces needed by sampling them from the characteristic set [9] of the task distribution, rather than at random.

We validate these two metrics and Kernel Representations with transfer RL experiments on a simple grid world domain, inspired by the video game Minecraft. In this world, the tasks consist in visiting cells in a suitable order, that is encoded as Linear Temporal Logic formulas evaluated on finite traces (LTLf) [10]. A frame from the environment, along with some examples of tasks, are shown in figure 1 (a-b). The results suggest a clear advantage of our Kernel Representation against the non-transfer-aware baseline given by Reward Machines [4]. At the same time, they confirm our hypothesis that our representation quality metrics can give precious insights about the transfer performance before any interaction with the environment.

We believe that this work could represent a solid foundation for more advanced ways to use the semantic similarity between temporal states to obtain lightweight and simple transferable tasks state representations, directly exploiting the task semantics, without the need to learn such representations from the environment. In other words, this is a first crucial step towards the long-standing challenge of making an artificial agent follow temporally-extended instructions without extensive training on every possible scenario, exploiting the semantics of the language in which the instructions are given, and nothing else.

2. Related Work

Extending classical RL to temporal goals requires augmenting the state of the MDP with some kind of representation of the current progress towards satisfying the goal. The goal itself is most commonly specified with LTL, as in geometric-LTL (G-LTL) [11], and Restraining Bolts [3]. In these cases, the current progress towards the goal is encoded as the state of an automaton equivalent to the LTL formula. Reward Machines (RMs) [12] [4] [13] follow the same principle, but they directly represent tasks as Finite State Automata.

In all of the above approaches, however, the representation of the current progress is specific to a particular task, meaning that the learned policies cannot, in general, be reused for other tasks. Early work on semantic-based transfer has focused on allowing transfer when a new task is entirely composed by situations already seen during training [14]. The main limitation of these kind of approach is that it cannot transfer to unseen temporal states that are not exactly equivalent to those seen during training. In contrast, our method allows to map semantically close temporal states to close points in the semantic space, generalizing to unseen situations.

More recent work, instead, focuses on fully transferable task state representations, such that a more general policy can be learned and transferred to entire task distributions. One direction is that of learning independent policies for various subtasks in the environment [15, 16, 17, 18, 19], which are then combined to solve more complex tasks. Another class of approaches is based on exploiting the syntactic structure of the task formula in the design of the agent Neural Network, either by using it to shape the policy networks [6], or by feeding it directly to a Graph Neural Network to learn a latent embedding [20]. However, none of these approaches makes use of a semantic representation of the task specification to make it transferable. Instead, in contrast to both compositional and NN-based methods, we explore this kind of approach, inspired by [7] and [8], which investigate how to generate semantic-preserving embeddings of logical formulae.

3. Background

Linear Temporal Logic and Deterministic Finite Automata Linear Temporal Logic (LTL) [21] is a language which extends traditional propositional logic with modal operators. With the latter we can specify rules that must hold *through time*. In this work, we use LTL interpreted over finite traces (LTLf) [10]. Such interpretation allows the executions of arbitrarily long traces, but not infinite, and is adequate for finite-horizon planning or RL problems. Given a set P of propositions, the syntax for constructing an LTLf formula ϕ is given by

$$\phi := \top \mid \bot \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid \phi_1 U \phi_2 \tag{1}$$

where $p \in P$. We use $\top$ and $\bot$ to denote true and false respectively. X (Next) and U (Until) are temporal operators. Other temporal operators are: N (Weak Next) and R (Release) respectively, defined as $N\phi \equiv \neg X \neg\phi$ and $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$; G (globally) $G\phi \equiv \bot R \phi$ and F (eventually) $F\phi \equiv \top U \phi$. A trace $x_p = x_p^{(0)}, x_p^{(1)}, ...$ is a sequence of propositional assignments, where $x_p^{(t)} \in 2^P$ $(t \geq 0)$ is the point of x_p a time t. Intuitively, $x_p^{(t)}$ is the set of propositions that are true at instant t. Additionally, $|x_p|$ represents the length of x_p. Since each trace is finite, $|x_p| < \infty$ and $x_p \in (2^P)^*$. If the propositional symbols in P are *mutually exclusive*, e.g. the domain produces one and only one symbol true at each step, $x_p^{(t)} \in P$ $(t \geq 0)$. In this work we assume this second setting of the domain, that is known as the Declare assumption [22]. We denote with $x_p \vDash \phi$, when x_p satisfy the formula ϕ. We refer the reader to [21] for a formal description of the operators' semantics. Any LTLf formula ϕ can be translated in an equivalent Deterministic Finite Automaton (DFA) $A_\phi = (\Sigma_\phi, Q_\phi, q_0, \delta_{t,\phi}, F_\phi)$ [10], where Σ_ϕ is the automaton alphabet, Q_ϕ is the set of states, $q_0 \in Q_\phi$ is the initial state, $\delta_{t,\phi} : Q_\phi \times \Sigma_\phi \to Q_\phi$ is the transition function and $F_\phi \subseteq Q_\phi$ is the set of final states. Σ_ϕ is equal to P or 2^P, depending on whether the Declare assumption is upheld. Thus, in our case, $\Sigma_\phi = P$. We define the transition function over strings $\delta^*_{t,\phi} : Q_\phi \times \Sigma^*_\phi \to Q_\phi$ recursively

$$\begin{aligned} &\delta^*_{t,\phi}(q, \epsilon) = q \\ &\delta^*_{t,\phi}(q, p + x_p) = \delta^*_{t,\phi}(\delta_{t,\phi}(q, p), x_p) \end{aligned} \tag{2}$$

Where $p \in P$ is a symbol, ϵ is the empty traces. $x_p \in \Sigma^*_\phi$ is a trace, and $p + x_p$ is the concatenation of p and x_p. Let $L(A_\phi)$ be the language composed by all the strings accepted by the A_ϕ we have $x_p \vDash \phi$ iff $x_p \in L(A_\phi)$.

Non-Markovian Reward Decision Processes and Reward Machines In RL the agent-environment interaction is generally modeled as a Markov Decision Process (MDP) [1]. An MDP is a tuple (S, A, t, r, γ), where S is the set of environment *states*, A is the set of agent's *actions*, $t : S \times A \times S \to [0, 1]$ is the *transition function*, $r : S \times A \to \mathbb{R}$ is the *reward function*, and $\gamma \in [0, 1]$ is the *discount factor* expressing the preference for immediate over future reward. In this classical setting, transitions and rewards are assumed to be Markovian – i.e., they are functions of the current state only. A decision process can be non-markovian because the markov property does not hold on the reward function $r : (S \times A)^* \to \mathbb{R}$, or the transition function $t : (S \times A)^* \times S \to [0, 1]$, or both. In this work we focus on Non-Markovian *Reward* Decision Processes (NMRDP) [23]. Learning an optimal policy in such settings is hard, since the current environment outcome depends on the entire history of state-action pairs the agent has explored from the beginning of the episode. Many works have investigated how to construct Markovian state representations of NMRDP, such for example Reward Machines (RMs). RMs are an automata-based representation of non-Markovian reward functions [24]. Given a finite set of propositions P representing abstract properties or events observable in the environment, RMs specify temporally extended rewards over these propositions while exposing the compositional reward structure to the learning agent. Formally, in this work we assume the reward can be represented as a Reward Machine $RM = (P, Q, R, q_0, \delta_t, \delta_r, L)$, where P is the automaton alphabet, Q is the set of automaton states, R is a finite set of continuous reward values, q_0 is the initial state, $\delta_t : Q \times P \to Q$ is

the transition function, $\delta_r : Q \times P \to R$ is the reward function, and $L : S \to P$ is the labelling function, which recognizes symbols in the environment states. Let $x_s = [s^{(1)}, s^{(2)}, ..., s^{(t)}]$ be a sequence of states the agent has observed in the environment up to the current time instant t, we define the labeling function over strings $L^* : S^* \to P^*$ as $L^*(x_s) = [L(s^{(1)}), L(s^{(2)}), ..., L(s^{(t)})]$. We denote with δ_t^* and δ_r^* the transition function and the reward over strings, which are defined recursively, analogously to Equation 2. Given x_s, the RM produces an history-dependent reward value at time t, $r^{(t)} = \delta_r^*(q_0, L^*(x_s))$ and an automaton state at time t, $q^{(t)} = \delta_t^*(q_0, L^*(x_s))$. The reward value can be used to guide the agent toward the satisfaction of the task expressed by the automaton, while the automaton state can be used to construct a Markovian state representation. In fact it was proven that the augmented state $(s^{(t)}, q^{(t)})$ is a Markovian state for the task expressed by the RM [3].

Semantic Similarity Function for Signal Temporal Logic Formulas Signal Temporal Logic (STL) [25] is an extension of LTL to *continuous variables* and *continuous time*. STL formulas are used to monitor properties of trajectories. A trajectory is a function $\xi : I \to D$, with a time domain $I \subseteq \mathbb{R}_{\geq 0}$, and a state space $D \subseteq \mathbb{R}^n$, for some $n \in \mathbb{N}$. We define the trajectory space $\mathcal{T}$ as the set of all possible continuous functions over D. An atomic predicate p of STL is a continuous computable predicate on $x \in \mathbb{R}^n$ of the form of $f(x_1, ..., x_n) \geq 0$. STL has a sintax similar to that of LTL defined in Equation 1, where the semantic of temporal operators is just slightly modified to take into account the continuity of time and input variables. In particular, in STL can be given not only the classic boolean notion of satisfaction, denoted by $s(\phi, \xi) = 1$ if ξ satisfies ϕ, and 0 otherwise, but also a quantitative one, denoted by $\rho(\phi, \xi)$, with $-1 \leq \rho(\phi, \xi) \leq 1$. This measures the quantitative level of satisfaction of a formula for a given trajectory, evaluating how "robust" is the satisfaction of ϕ with respect to perturbations in the signal [26]. In [7] the authors define a kernel function approximating the *semantic* distance between two STL formulas ϕ and ψ as:

$$k(\phi, \psi) = \int_{\xi \in \mathcal{T}} \rho(\phi, \xi)\rho(\psi, \xi) d\mu_0(\xi) \tag{3}$$

Note that the integral is defined over the entire trajectory space $\mathcal{T}$. Integrating over uncountably many trajectories requires to put a finite measure on them, according to which to integrate. Therefore μ_0 is a probability measure over trajectories which prefers "simple" trajectories, where the signals do not change too dramatically or, in other words, the total variation is low. Notice that $k(\phi, \psi)$ depends on the *language* of the two formulas, assigning high similarity for couples of formulas that are satisfied by the same (or similar) set of trajectories. This similarity function will be our starting point to define a transferable semantic representation state-space for NMRDPs.

4. Method

Given a certain temporally extended task expressed as an LTLf formula ϕ, we aim to define a new representation of the task's temporal states, $\alpha_\phi(q)$, with $q \in Q_\phi$, which can be transferred to new temporal tasks executed in the same environment. We consider the temporal states of the task as the states of the automaton obtained by translating ϕ into the DFA A_ϕ.

In particular, after training the agent on a certain set of training formulas Φ_{train}, we test it on a set of test tasks Φ_{test} without acquiring any experience for these new tasks in a zero-shot transfer learning experiment. More formally, we consider NMRDPs constructed as the combination of a transition system (an MDP without reward) and a reward expressed as an RM, $NMRDP = (S, A, t, RM_\phi, \gamma)$. Let P be a finite set of observable symbols in the environment via the labeling function L, $RM_\phi = (P_\phi, Q_\phi, R = \{0, 1\}, q_0 = 0, \delta_{t,\phi}, \delta_{r,\phi}, L)$, where

$$\delta_{r,\phi}(q) = \begin{cases} 1 & \text{if } q \in F_\phi \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and we assume that $P_\phi \subseteq P \; \forall \phi \in \Phi_{train} \cup \Phi_{test}$. Notice that: (i) the labeling function is the same for all reward machines, as it depends on the environment rather than the task; (ii) we assume a binary reward

function ($R = \{0, 1\}$) for all tasks; in particular, we assume $\delta_{r,\phi}$ takes the value 1 in all final states of A_ϕ and 0 in the other states of the automaton.

As is customary in non-Markovian-reward RL [4, 27], we condition the policy on *both* the environment raw state and the *temporal* state in order to build a markovian state representation. We therefore train a policy $\pi : S \times T \to A$, returning an action for the couple (environment state, temporal state), where $T \subseteq \mathbb{R}^B$ is the *semantic temporal space* that we will define in the next section.

4.1. Kernel Representation for LTLf Tasks

We aim to define a semantic metric space for mapping the temporal states of LTLf formulas, where the metric measures the semantic distance between two temporal states. More formally, we denote with $A_{\phi,q}$ the automaton generated by the formula ϕ, with the initial state q: $A_{\phi,q} = (P, Q_\phi, q_0 = q, \delta_{t,\phi}, F_\phi)$, and we denote its language by $L_{\phi,q}$. Let q and q' be two temporal states generated by two formulas ϕ and ϕ', respectively (which could also be the same formula). Ideally, we want the distance between the two points in T, $\alpha_\phi(q)$ and $\alpha_{\phi'}(q')$, to be as small as possible when the languages $L_{\phi,q}$ and $L_{\phi',q'}$ are similar.

$$dist(\alpha_\phi(q), \alpha_{\phi'}(q')) \simeq dist(L_{\phi,q}, L_{\phi',q'}) \tag{5}$$

The representation space above could be approximated by constructing an embedding space. However, to avoid computing the distance between all combinations of states generated by all considered formulas, we choose a different solution. Specifically, we construct the representation in terms of some formulas that will serve as *base* of the metric space. Thus, we select a set of B LTLf $\Phi_{base} = \{\phi_{b_1}, \phi_{b_2}, ..., \phi_{b_B}\}$, and we define the *Kernel Representation* $\alpha_\phi(q)$ as the continuous vector with B components.

$$\alpha_\phi(q) = [dist(L_{\phi_{b_1},q_0}, L_{\phi,q}), ..., dist(L_{\phi_{b_B},q_0}, L_{\phi,q})] \tag{6}$$

The base formulas are fixed before the training phase and must remain the same during the test phase to ensure transferability. Now, let us define the semantic distance between two languages by adapting the similarity function in Equation 3 to our case.

$$\begin{aligned} dist(L_{\phi,q}, L_{\phi',q'}) &= 1 - sim(L_{\phi,q}, L_{\phi',q'}) \\ sim(L_{\phi,q}, L_{\phi',q'}) &= \frac{1}{|\bar{P}|} \sum_{x_p \in \bar{P} \subseteq P^*} \mathbf{1}\Big\{(x_p \in (L_{\phi,q} \cap L_{\phi',q'}) \vee (x_p \notin (L_{\phi,q} \cup L_{\phi',q'}))\Big\} \end{aligned} \tag{7}$$

The similarity between two temporal states q and q' generated respectively by two formulas ϕ and ϕ' is higher when the number of traces in the set $\bar{P}$ that belong to *both* or *none* of the languages $L_{\phi,q}$ and $L_{\phi',q'}$ is higher. Notice that the similarity value is 1 if the two languages contain exactly the same traces. Moreover, we replace the integral from Equation 3 with a summation, since in our case the traces are discrete in both value and time. For the same reason, we do not weigh the importance of a trace x_p in the sum by the probability $\mu_0(x_p)$ preferring low total variation traces because this probability function does not favor any of the *digital* signals we are interested in. Therefore, we decided to limit the summation to a *finite* set of traces $\bar{P} \subseteq P^*$, with $|\bar{P}| < \infty$, and weigh each trace uniformly. This simplifies the calculation of similarity but also makes the similarity function in Equation 7 only an *approximation* of the true similarity between the two languages, which is more faithful the more $\bar{P}$ resembles P^*. We investigate in experiments how different choices of $\bar{P}$ influence the performance of the semantic representation for RL transfer. Last but not least, note that while the similarity function in Equation 3 is defined between two formulas, our similarity function is defined between two formula-state pairs. This extension is necessary to apply the representation to non-Markovian RL, where knowing the formula alone is not enough, but we want to know at each time instant the current temporal state of the task. In the next section we will discuss how to practically set the hyperparameters of our representation to better transfer the policy between multiple temporal tasks.

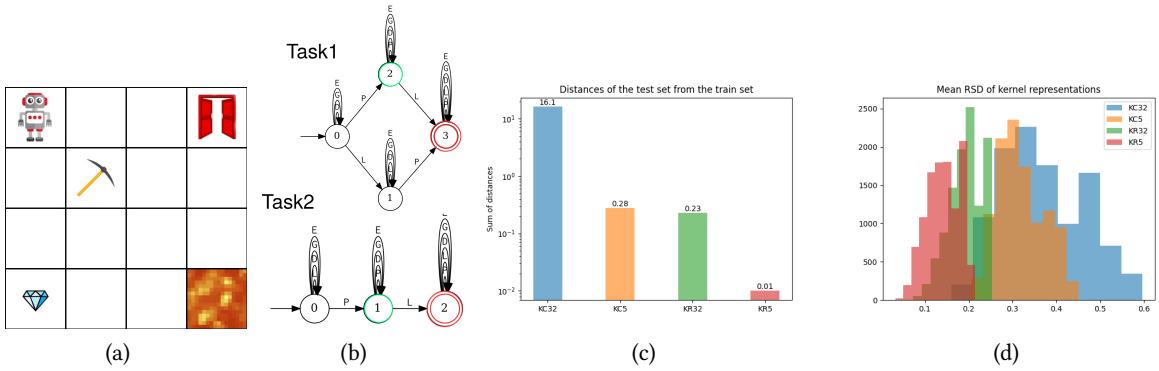

Figure 1: (a) Minecraft-like environment for temporally extended navigation tasks. (b) The automata corresponding to two tasks of example. T1: "reach the pickaxe (P) and the lava cell (L) in whatever order", T2: "go to the pickaxe and after that reach the lava". Note that state 2 of T1 and state 1 of T2 (colored in green) are semantically equivalent temporal states, as state 3 of T1 and state 2 of T2 (colored in red). (c) Sums of inter-task variation between test and train tasks for different Kernel representation. (d) Distribution of the intra-task variation for different Kernel Representations among all sets.

4.2. Representation Hyperparameters

Given the definition of our semantic temporal representation for regular tasks, (Equations 6 and 7), notice that this depends on two reference choices: (i) the set of *reference automata* Φ_{base}, (ii) the set of *reference traces* $\bar{P}$. The selection of these references can significantly impact the quality of the representation. In our investigation, we explore two settings for the representation, each corresponding to different strategies for sampling reference automata and traces: (i) a training-formulas-*independent* representation , (ii) a training-formulas-*dependent* representation

In the first setting, we operate without any prior knowledge about the environment or the training tasks. Here, we define the set of reference traces $\bar{P}$ as the complete set of traces with a maximum length of l_{max}, denoted as $P^{l_{max}}$. Note that the number of traces in $P^{l_{max}}$ grows exponentially with the maximum length. Therefore, setting a high value for l_{max} is computationally impractical. However, this remains the best sampling strategies for simple tasks, that can be captured by short traces. For the selection of reference automata, we opt for a random sampling approach, generating B random automata as a base, following a method similar to that proposed by [28] to select non trivial automata of a target size $|Q|$. This first representation benefits from being *completely independent* by both the tasks chosen and the environment, however has as limitations that can struggle to capture complex tasks because the reference traces may be too short to capture long-time-dependencies and/or because random automata are too far from the languages of the training tasks.

In the second setting, we leverage knowledge of at least the training formulas Φ_{train} to tailor the representation hyperparameters to better suit the temporal states of interest. Despite this, we do not assume any knowledge about the test tasks or the operating environment. For selecting reference traces in this scenario, we adopt the concept of the *characteristic set* of a DFA from the literature on automata induction [29]. The characteristic set of an automaton A, denoted as $\mathscr{C}(A)$, is the minimal set of labeled traces necessary to accurately identify the DFA from traces [9]. In particular, we calculate the characteristic set of each training automata A_ϕ, with $\phi \in \Phi_{train}$, with algorithm [30], and we choose reference traces from these sets. In this way we know that each trace in $\bar{P}$ is *necessary* to capture at least one state of one training automaton. Again, since the union of all the characteristic sets of training tasks may result in a very large set of traces, to reduce the computational cost, we randomly sample in this set a certain number of reference traces, without considering them all, having therefore $\bar{P} \subseteq \bigcup_{\phi \in \Phi_{train}} \mathscr{C}(A_\phi)$. Regarding the choice of reference automata, we simply randomly select B training task formulas, $\Phi_{base} \subseteq \Phi_{train}$. We will show in the next section how these strategies for the reference selection will impact on the learning and transfer process.

5. Experiments

In this section we report some preliminary experiments validating our framework. We publicly provide our code on GitHub at https://github.com/KRLGroup/kernel-representations-transfer/.

Environment We test our approach on temporally extended navigation tasks. The latter are accomplished in a Minecraft-like environment, similar to those considered by [31] and [32]. The environment consists in a grid world containing different items like the one depicted in Figure 1 (a). The RM input alphabet is composed by a symbol for each different item plus a symbol indicating the absence of items (empty-cell). Each symbol is considered set to True when the agent is in the item-cell and false otherwise. The agent has to learn how to navigate the grid in order to satisfy the temporal constraint specified in the task. Each task is expressed in LTLf and then translated into an automaton using the tool LTL2DFA [33]. The reward given to the agent is 100 when it reaches a final DFA state and 0 otherwise.

Task dataset A total of 11000 formulas were used for the transfer learning experiments, divided into: (i) 9000 for training; (ii) 1000 for validation; (iii) 1000 for testing. All 11000 formulas were sampled without repetition from a distribution of partially-ordered tasks, as described in [5]. These tasks consist of multiple sequences of propositions, where the propositions in each sequence must be solved in order, but the sequences can be solved in parallel. An example of this kind of taks (specified informally in English) is: "*go to the pickaxe (P), the gem (G), and the door (D) in that order, and reach the lava (L) and the gem (G) in that order*" where one valid solution would be to reach lava, pickaxe, gem, door in that order. The number of possible unique tasks that can be generated by this distribution is in the order of 10^{39} [5].

Configurations The experiments were performed using 3 different variants of the Kernel Representation and a baseline representation. These differ only in the way the observation over the current state of the DFA is represented, as follows: (**KR5**) and (**KR5**) are generated respectively using 5 and 32 randomly sampled DFAs with 10 states, over all possible traces of length in $[1, 5]$; (**KC5**) and (**KC32**) are generated respectively using 5 and 32 DFAs sampled from the training distribution, over a subset of the union of all the characteristic sets of the training set DFAs; (**VRM**) is the Vanilla Reward Machines baseline representation, i.e. an integer indicating the current state of the DFA, with 0 always being the initial state. For KC5 and KC32, the set of traces used to compute the Kernel Representation was generated as follows: (i) first, the union of all the characteristic sets of all DFAs in the training set was computed; (ii) from this set, we removed all traces that were subtraces of others already contained in the set; (iii) we truncated this set to the first N traces such that the sum of their lengths was as close as possible to 20000 characters. This resulted in a set of $N = 1611$ characteristic traces of total length 20008. Note that the other two representations, KR5 and KR32, are computed using trace sets of much greater size, but similar total length. In fact, the number of all possible traces with length in $[1, 5]$ is $5^1 + 5^2 + \cdots + 5^5 = 3905$, with their total length being $5^1 + 2 \cdot 5^2 + \cdots + 5 \cdot 5^5 = 18555$.

5.1. Results

Offline Analysis of Kernel Representations Before delving into the RL and transfer experiments, in this section we analyze the four generated Kernel representations and their potential informative power. We believe this is especially useful to compare the different hyperparameters used to generate them. In particular, we investigate two desirable properties of approximate Kernel representations: *intra-task variation*, i.e. different states of the same DFA should be represented by different vectors, since they have different semantics; *inter-task variation*, i.e. states in the test DFAs that have semantics similar to states in a train DFA, should be represented by close vectors. While these hold for an ideal Kernel representation, there is no guarantee for approximations, which motivates the need for this kind of analysis. Finally, notice that ideally we desire representations with both *high intra-task variation* and *low inter-task variation.*

Table 1
Percentage of solved formulas and discounted return on the test and training sets, and number of training steps needed to finish the training curriculum. Results are averaged over 5 runs.

Conf.	% Solved (train)	Disc. ret. (train)	% Solved (test)	Disc. ret. (test)	Train Steps (M)
KR5	93.1 ± 1.5	30.1 ± 1.2	92.3 ± 2.2	28.9 ± 1.7	20.2 ± 3.4
KR32	**94.3 ± 1.3**	**31.1 ± 1.4**	**94.1 ± 1.9**	**30.1 ± 1.6**	22.9 ± 3.5
KC5	93.9 ± 1.9	30.6 ± 1.4	93.9 ± 2.3	29.6 ± 1.6	**14.9 ± 0.8**
KC32	92.3 ± 2.0	29.6 ± 1.2	92.2 ± 2.3	28.6 ± 1.3	24.9 ± 3.5
VRM	90.5 ± 0.0	28.7 ± 0.0	90.4 ± 0.0	27.5 ± 0.0	32.4 ± 3.6

In our experiments, we define intra-task variation of a task A_ϕ as the relative standard deviation (RSD) among the representations of all its states. We also define inter-task variation of a task A_ϕ with respect to a set of k tasks $\Gamma = \{A_{\phi_1}, ..., A_{\phi_k}\}$ as the sum of the minimum distances between the representations of its states and all the representations of all states of the DFAs in the set Γ, namely as $\sum_{q \in Q_\phi} min_{q' \in \cup_{i=1}^{i=k} Q_{\phi_i}} dist(\alpha_\phi(q), \alpha_{\phi i}(q'))$. Results are reported in Figure 1 (c) and (d). These highlight 3 different cases: low variations (KR5), balanced (KR32 and KC5), and high variations (KC32). Thus, representations with extremely good intra-task variation sacrifice inter-task variation, and vice versa. However, extremely low inter-task variation is not beneficial when intra-task variation is also low (KR5), since this could mean that semantically different temporal states collapse into too close vectors. Vice versa, an extremely high intra-task variation is not beneficial when inter-task variation is also high (KC32), since it could mean that semantically close temporal states are mapped to distant vectors, hindering transfer learning. Thus, these results suggest that the best representations are those that strike a balance between intra and inter-task variation (KR32 and KC5), such that they do not sacrifice any of the two properties.

Transfer Learning Performance To evaluate the transfer learning performance, we train a PPO [34] agent on the training set for each of the configurations above, recording the policy with the best validation performance during training. We use a curriculum-learning training scheme, sorting tasks in the training set by the number of states of the corresponding DFA as a proxy of task difficulty. Moreover, we train the agent on the same task until either it is solved, or a maximum number of episodes were seen by PPO before moving on to the next one. After training, we evaluate the zero-shot transfer performance of the policy on the test tasks. Table 5.1 reports the percentage of training and test task solved, the discounted return obtained, and the number of training steps needed to finish the curriculum. KR32 obtains the best results in all 3 performance metrics, with KC5 being a close second, and VRM always obtains the worst results. This suggests that the careful choice of traces and reference automata allowed KC5 to reach results comparable with KR32, a representation that uses a similar total trace length and ~ 6 times the number of reference automata, again highlighting the importance of these aspects. Moreover, KC5 took the least amount of training steps to finish the curriculum among all representations, which is less than half of that required by the VRM baseline. This suggests that a careful choice of traces and references is also highly beneficial for sample efficiency. Moreover, these results confirm the findings of the offline representation analysis, in that representations with extreme variations (KR5 and KC32) obtain worse performance than the balanced representations (KC5 and KR32).

6. Conclusions and Future Work

In this paper we introduce a novel *Kernel Representation* for temporal states of LTL tasks, to address the challenge of policy transfer in temporally-extended Reinforcement Learning. We show that conditioning the agent policy over the semantic temporal space is beneficial for transferring the policy to *new* LTL instructions, even if this space is computed *offline* and does not leverage any learning module. We investigate how the choice of reference traces and automata impacts the learning and transfer

performances and the quality of the representation, defining two metrics (*intra* and *inter-task variation*) to analyze the representation offline. We speculate that representations that strike a balance between these two metrics have better chances of successful transfer to unseen tasks, and confirm this hypothesis in our RL experiments. In the future we would like to develop a strategy to select reference and train tasks so as to maximize the probability of transfer to a broader set of test formulas.

Acknowledgments

This work has been partially supported by PNRR MUR project PE0000013-FAIR.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, second ed., The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[2] M. L. Littman, U. Topcu, J. Fu, C. L. I. Jr., M. Wen, J. MacGlashan, Environment-independent task specifications via GLTL, CoRR abs/1704.04341 (2017). URL: http://arxiv.org/abs/1704.04341. arXiv:1704.04341.

[3] G. De Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications, Proceedings of the International Conference on Automated Planning and Scheduling 29 (2021) 128–136. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/3549.

[4] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Reward machines: Exploiting reward function structure in reinforcement learning, J. Artif. Intell. Res. 73 (2022) 173–208. URL: https://doi.org/10.1613/jair.1.12440. doi:10.1613/JAIR.1.12440.

[5] P. Vaezipoor, A. C. Li, R. T. Icarte, S. A. McIlraith, Ltl2action: Generalizing LTL instructions for multi-task RL, in: Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, 2021, pp. 10497–10508. URL: http://proceedings.mlr.press/v139/vaezipoor21a.html.

[6] Y.-L. Kuo, B. Katz, A. Barbu, Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of ltl formulas, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 5604–5610.

[7] L. Bortolussi, G. M. Gallo, J. Kretínský, L. Nenzi, Learning model checking and the kernel trick for signal temporal logic on stochastic processes, in: Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I, 2022, pp. 281–300. URL: https://doi.org/10.1007/978-3-030-99524-9_15. doi:10.1007/978-3-030-99524-9_15.

[8] G. Saveri, L. Bortolussi, Towards invertible semantic-preserving embeddings of logical formulae, 2023. arXiv:2305.03143.

[9] E. M. Gold, Complexity of automaton identification from given data, Information and Control 37 (1978) 302–320. URL: https://www.sciencedirect.com/science/article/pii/S0019995878905624. doi:https://doi.org/10.1016/S0019-9958(78)90562-4.

[10] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, p. 854–860.

[11] M. L. Littman, U. Topcu, J. Fu, C. Isbell, M. Wen, J. MacGlashan, Environment-independent task specifications via gltl, arXiv preprint arXiv:1704.04341 (2017).

[12] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Ltl and beyond: Formal languages for reward function specification in reinforcement learning., in: IJCAI, volume 19, 2019, pp. 6065–6073.

[13] R. T. Icarte, T. Klassen, R. Valenzano, S. McIlraith, Using reward machines for high-level task specification and decomposition in reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 2107–2116.

[14] R. Toro Icarte, T. Q. Klassen, R. Valenzano, S. A. McIlraith, Teaching multiple tasks to an RL agent using LTL, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2018, pp. 452–461.

[15] B. G. León, M. Shanahan, F. Belardinelli, Systematic generalisation through task temporal logic and deep reinforcement learning, Adaptive and Learning Agents Workshop at International Conference on Autonomous Agents and Multiagent Systems (2022).

[16] B. G. León, M. Shanahan, F. Belardinelli, In a nutshell, the human asked for this: Latent goals for following temporal specifications, in: International Conference on Learning Representations (ICLR), 2022.

[17] B. Araki, X. Li, K. Vodrahalli, J. Decastro, M. Fry, D. Rus, The logical options framework, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of *Proceedings of Machine Learning Research*, PMLR, 2021, pp. 307–317. URL: https://proceedings.mlr.press/v139/araki21a.html.

[18] J. Andreas, D. Klein, S. Levine, Modular multitask reinforcement learning with policy sketches, in: International Conference on Machine Learning, PMLR, 2017, pp. 166–175.

[19] J. X. Liu, A. Shah, E. Rosen, G. Konidaris, S. Tellex, Skill transfer for temporally-extended task specifications, 2023. arXiv:2206.05096.

[20] P. Vaezipoor, A. C. Li, R. A. T. Icarte, S. A. Mcilraith, LTL2action: Generalizing LTL instructions for multi-task RL, in: International Conference on Machine Learning, PMLR, 2021, pp. 10497–10508.

[21] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 46–57. URL: https://doi.org/10.1109/SFCS.1977.32. doi:10.1109/SFCS.1977.32.

[22] G. De Giacomo, R. De Masellis, M. Montali, Reasoning on ltl on finite traces: Insensitivity to infiniteness, Proceedings of the AAAI Conference on Artificial Intelligence 28 (2014). URL: https://ojs.aaai.org/index.php/AAAI/article/view/8872. doi:10.1609/aaai.v28i1.8872.

[23] F. Bacchus, C. Boutilier, A. Grove, Rewarding behaviors, Portland, OR, 1996, pp. 1160–1167. URL: behaviors.pdf.

[24] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, S. 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, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 6065–6073. URL: https://doi.org/10.24963/ijcai.2019/840. doi:10.24963/ijcai.2019/840.

[25] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, in: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings, 2004, pp. 152–166. URL: https://doi.org/10.1007/978-3-540-30206-3_12. doi:10.1007/978-3-540-30206-3_12.

[26] A. Donzé, T. Ferrère, O. Maler, Efficient robust monitoring for STL, in: Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings, 2013, pp. 264–279. URL: https://doi.org/10.1007/978-3-642-39799-8_19. doi:10.1007/978-3-642-39799-8_19.

[27] G. D. Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications, 2019. arXiv:1807.06333.

[28] I. Zakirzyanov, A. A. Shalyto, V. I. Ulyantsev, Finding all minimum-size dfa consistent with given examples: Sat-based approach, in: SEFM Workshops, 2017.

[29] C. de la Higuera, A bibliographical study of grammatical inference, Pattern Recognition 38 (2005) 1332–1348. URL: https://www.sciencedirect.com/science/article/pii/S0031320305000221. doi:https://doi.org/10.1016/j.patcog.2005.01.003, grammatical Inference.

[30] P. García, D. López, M. Vázquez de Parga, Polynomial characteristic sets for dfa identification, Theoretical Computer Science 448 (2012) 41–46. URL: https://www.sciencedirect.com/science/article/pii/S0304397512004070. doi:https://doi.org/10.1016/j.tcs.2012.04.042.

[31] J. Corazza, I. Gavran, D. Neider, Reinforcement learning with stochastic reward machines, Proceedings of the AAAI Conference on Artificial Intelligence 36 (2022) 6429–6436. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20594. doi:10.1609/aaai.v36i6.20594.

[32] A. C. Li, Z. Chen, P. Vaezipoor, T. Q. Klassen, R. T. Icarte, S. A. McIlraith, Noisy symbolic abstractions for deep RL: A case study with reward machines, CoRR abs/2211.10902 (2022). URL: https://doi.org/10.48550/arXiv.2211.10902. doi:10.48550/arXiv.2211.10902. arXiv:2211.10902.

[33] F. Fuggitti, Ltlf2dfa, 2019. doi:10.5281/zenodo.3888410.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, ArXiv abs/1707.06347 (2017). URL: https://api.semanticscholar.org/CorpusID:28695052.