Do it for HER: First-order Logic Reward Specification in Reinforcement Learning

Pierriccardo Olivieri
Politecnico di Milano
pierriccardo.olivieri@polimi.it

Fausto Lasca
Politecnico di Milano
fausto.lasca@mail.polimi.it

Alessandro Gianola
INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
alessandro.gianola@tecnico.ulisboa.pt

Matteo Papini
Politecnico di Milano
matteo.papini@polimi.it

Abstract

In this work, we propose a novel framework for the logical specification of non-Markovian rewards in Markov Decision Processes (MDPs) with large state spaces. Our approach leverages Linear Temporal Logic Modulo Theories over finite traces (LTLf$^{\text{MT}}$), a more expressive extension of classical temporal logic in which predicates are first-order formulas of arbitrary first-order theories rather than simple Boolean variables. This enhanced expressiveness enables the specification of complex tasks over unstructured and heterogeneous data domains, promoting a unified and reusable framework that eliminates the need for manual predicate encoding. However, the increased expressive power of LTLf$^{\text{MT}}$ introduces additional theoretical and computational challenges compared to standard LTLf specifications. We address these challenges from a theoretical standpoint, identifying a fragment of LTLf$^{\text{MT}}$ that is tractable but sufficiently expressive for reward specification in an infinite-state-space context. From a practical perspective, we introduce a method based on reward machines and Hindsight Experience Replay (HER) to translate first-order logic specifications and address reward sparsity. We evaluate this approach to a continuous-control setting using Non-Linear Arithmetic Theory, showing that it enables natural specification of complex tasks. Experimental results show how a tailored implementation of HER is fundamental in solving tasks with complex goals.

1 Introduction

In Reinforcement Learning (RL), an agent interacts with an environment and receives a scalar reward signal as feedback, which it uses to improve its decision-making over time. In principle, under the *reward hypothesis* [Sutton et al., 1998], any task can be modeled by specifying an appropriate reward function. However, reward engineering is a difficult task, often heuristic in nature and reliant on domain knowledge. Furthermore, many tasks of interest are inherently non-Markovian, making commonly used Markovian reward functions less effective for learning. Introducing structure in the reward via logic specification helps address these challenges by bringing reward engineering closer to human language and improving interpretability. In *logic reward specification*, the reward the agent receives at each time step depends on whether a formula is satisfied based on the set of true propositional symbols at that time step. This approach allows expressing complex tasks using *temporal logic*. To better picture this, consider a robot operating in a warehouse, having to deliver a box to a specific location. This task has two sequential sub-goals: (A) picking up the box in a given position (x_o, y_o), then (B) delivering it to the designated location (x_g, y_g). This behavior can be enforced via a logic specification that encodes the temporal relationship between events A and B, such as "Do A, then B".

Currently, most works in this line employ logic specification based on Linear Temporal Logic with finite traces (LTLf) [De Giacomo and Vardi, 2013, Brafman et al., 2018]. Stemming from its counterpart with infinite traces (LTL [Pnueli, 1977a]), which is the most common language for specifying system properties in the field of formal verification. In the last decade, LTLf has also gained popularity in AI, where reasoning over finite executions, such as in planning tasks, is more intuitive [Giacomo and Rubin, 2018]. LTLf offers sufficient expressiveness for many *discrete* domains, and benefits from a solid toolchain for translating such specifications into automata or "reward machines" [Icarte et al., 2018]. However, the expressiveness of LTLf is limited to Boolean predicates, which constrain the complexity of the reward specification. Evaluating the truth of these predicates typically requires defining *encoding functions* manually. This limitation is often not an issue in discrete domains or for simple reward functions. However, in many real-world scenarios, such as business process management [Gianola et al., 2024]or robotics [Li et al., 2017], a richer reward specification *language* allows providing equally or more informative feedback to the learner *with less human intervention* and can make the specification of new tasks for the same environment significantly faster. The importance of an expressive specification language is even more evident in the presence of *heterogeneous* data types, as shown in the following extended example.

Warehouse robot example. The robot must reach the object's position, identify a specific object by its ID, and satisfy a weight constraint. The resulting predicate might be "$A = (x - x_o)^2 + (y - y_o)^2 < r^2 \wedge i = \text{“}\texttt{H123}\text{”} \wedge weight < 10$". The robot has a *heterogeneous* state described by variables: $x, y, w \in \mathbb{R}^+$, representing position and object weight ($w = 0$ if no object is being carried) and $i \in \{\texttt{H123}, \texttt{S456}, \ldots\}$ is a given set of IDs.

Typically, evaluating such a predicate requires some hand-crafted encoders, one to determine when the Euclidean distance between the agent's position (x, y) and the objects location (x_o, y_o) falls below a threshold r, and another to check the ID and find the object's weight in an inventory database. Even in this relatively simple example, implementing the specification involves (i) manually encoding each condition, and likely (ii) producing a specification that is not reusable across different tasks. Without such user-defined encodings, handling continuous or heterogeneous states is impossible in LTLf, which operates at the propositional level.

To overcome these limitations, we propose using Linear Temporal Logic Modulo Theories over finite traces ($LTLf^{MT}$) [Geatti et al., 2022, 2023, Faella and Parlato, 2024, Rodríguez and Sánchez, 2024, 2023, Rodríguez et al., 2025], a extension of LTLf in which the atomic propositions are replaced by first-order formulas interpreted over (combinations of) arbitrary first-order theories. LTLf$^{\text{MT}}$ enables reasoning about reward specifications using diverse first-order theories such as arithmetic over the reals and/or integers, uninterpreted functions and relations (to represent, e.g., relational databases [Ghilardi et al., 2023],[Gianola, 2023]), complex datatypes (e.g., lists, arrays) and even custom domain-specific predicates, providing the expressiveness needed to naturally encode more complex specifications. From a practical perspective, instead of manually defining separate encoders for each condition, we propose a unified framework that requires only a formula for the task and an assignment of values to constants. The formula can be expressed using different theories and evaluated via a universal Satisfiability Modulo Theories (SMT) solver [Barrett et al., 2021], without the need to explicitly encode it as a propositional predicate. Continuing with the delivery robot example, the predicate A discussed earlier can be expressed and evaluated in LTLf$^{\text{MT}}$ using the (non-linear) arithmetic theory over the reals, and evaluated via a standard SMT solver without the need for hand-crafted, domain-dependent encodings. By shifting the complexity to the logical specification layer, this framework allows for the definition of more expressive behaviors than LTLf, while maintaining generality and reusability across different tasks and domains though the use of off-the-shelf SMT solvers [Barbosa et al., 2022, de Moura and Bjørner, 2008]. Theoretically, the LTLf$^{\text{MT}}$ framework can be viewed as a composable method for reward specification where each theory acts as a building block, and adding more theories (or combinations thereof) increases the expressiveness of the framework. If no additional theories are included beyond Boolean predicates, we recover the standard LTLf semantics. To achieve greater expressivity, we can incorporate and combine arbitrary theories such as arithmetic and database theories.However, the choice of theories added to LTLf$^{\text{MT}}$ (some theories may lack available solvers) and some advanced features of the full logic (such as quantifiers and variable-lookahead operators) affect both decidability and solver support, impacting the framework's efficiency and practical usability. It is thus important to identify sound but sufficiently expressive fragments of the logic and well-supported theories that can handle the data types characteristic of the domain of interest.

Contributions and paper structure. After discussing related works in Section 2 and introducing the necessary technical background in Section 3, we present *two main contributions*— In **Section 4**, we define a theoretical framework for reward specifications by extending the classical LTLf logic to the more expressive LTLf$^{\text{MT}}$, in which atomic propositions are replaced with first-order formulas over arbitrary theories. Then, we identify a fragment of LTLf$^{\text{MT}}$ that (i) is free from all decidability issues other than those introduced by the theory itself (ii) allows for easy translation of formulas to automata, and yet (iii) is expressive enough to capture most goals of practical interest— In **Section 5** we apply the proposed framework when the integrated theory is Non-Linear Real Arithmetic Theory (NRA) to handle reward specifications in a continuous-control scenario. In this domain, we address the problem of reward sparsity that is typical of logic specifications, exploiting the intrinsically "structured" nature of goals in our first-order framework. We propose a solution that combines Counterfactual Experiences for Reward Machines (CRM) and Hindsight Experience Replay (HER) [Andrychowicz et al., 2017] in a nontrivial way. Empirical results on continuous control tasks with complex goals suggest an improvement w.r.t. using only CRM (Section 6).

2 Related Works

We discuss related works in terms of the two key challenges addressed in this paper: dealing with non-boolean data and dealing with reward sparsity.

Propositional languages equipped with labeling functions. Given the intrinsic "discrete" nature of LTL-based rewards, most works focus on MDPs with finite states [Bacchus et al., 1996, Thiébaux et al., 2002, Gretton et al., 2003, 2011, Gretton, 2014, Brafman et al., 2018, Bozkurt et al., 2020]. There are notable exceptions: Cai et al. [2021] consider continuous MDP states, but logic formulas are defined using propositional letters corresponding to higher level concepts such as "regions" of the state space. Their definition is extra-logical and left to the user's implementation. This approach is more explicit in the framework of "restraining bolts" [De Giacomo et al., 2019, 2020a,b] in terms of "features" mapping properties of the "world" (not necessarily observed by the agent) to propositional atoms. When this mapping is applied directly to the states of the MDP, it is usually called *labeling function* [Lacerda et al., 2015, Hasanbeig et al., 2020, 2023, Jiang et al., 2021].

Reward Machines (RM) [Icarte et al., 2018, 2022] are a programmatic alternative to logic specification where the user directlu builds a finite-state automaton. In fact, we can identify the automaton generated by an LTLf formula as a type of RM called *simple reward machine* with a binary reward function. As in many works on logic discussed above, the mapping between states of the MDP and propositional letters must be implemented separately and is also called a labeling function in this context.

The use of labeling functions has allowed the application of both logically-specified rewards and reward machines to MDPs with continuous states (and actions), opening the door to *deep* RL experiments. [e.g., Hasanbeig et al., 2020, Li et al., 2024]. However, in all the aforementioned examples, the labeling function (or equivalent) is treated as a black box. This approach complicates generalization between tasks, is prone to errors, and undermines the interpretability of the logical formulas. A notable exception is the temporal logic proposed by Li et al. [2017] for robotic control, which allows first-order terms of the kind $f(s) < c$, where s is a continuous state of the MDP, c a constant and f a function. We will similarly argue that inequality terms of this kind are enough to encode most goals of interest for the setting of continuous control. However, the first-order terms of Li et al. [2017] are mere propositional atoms for which the form of the labeling function is prescribed in advance. We will show in which sense LTLf$^{\text{MT}}$ is more versatile.

Existing remedies to sparse rewards. By definition, goals specified by means of logical formulas tend to generate *sparse* (i.e., rare) rewards. These are notoriously hard to optimize in RL and led to the early development of general-purpose *reward shaping* strategies [Dorigo and Colombetti, 1994, Ng et al., 1999]—the addition of extra feedback to guide the agent towards the goal. Within the recent *deep RL* literature, in particular regarding robotics applications, Hindsight Experience Replay (HER) [Andrychowicz et al., 2017] has established itself as a powerful alternative, if not the standard approach to goal-based tasks. However, HER works by exploiting the state-generalization capabilities of neural networks applied to goals defined as states or functions of state variables. This shared structure of goals and states is typically absent from specifications based on propositional logics or finite-state automata, or hidden inside black-box components such as labeling functions. The problem of sparsity was explicitly addressed in the RM literature, where several ad-hoc approaches were already proposed by [Icarte et al., 2022], namely Counterfactual experiences for Reward Machines (CRM), Hierarchical approaches (HRM), and Automated Reward Shaping (ARS). They also suggested to explore synergies between RM and HER, but to the best of our knowledge the idea was not explored further. Hierarchical approaches, in the form of temporal abstractions, can be found also in the LTL literature [Araki et al., 2021]. Notably, Jothimurugan et al. [2019] jointly propose a (propositional) reward-specification language and a reward-shaping method.

3 Preliminaries

In this section, we review the main concepts of RL and reward specification based on temporal logic.

Reinforcement learning. The reinforcement learning problem [Sutton et al., 1998] is usually defined as the solution to a Markov Decision Process (MDP) [Puterman, 1990]. An MDP consists of a tuple $\langle S, A, R, P, \gamma \rangle$ where S is the set of states, A is the set of actions and $R : S \times A \to \mathbb{R}$ is the reward function. The environment's dynamics are described by the transition function $P : S \times A \times S \to [0, 1]$, which represents the probability of transitioning to state s' given we are in state $s \in S$ and take action $a \in A$. The discount factor $\gamma \in [0, 1)$ weighs the importance of future rewards versus immediate ones. The agent acts according to a policy $\pi : S \to \Delta(A)$, which assigns a probability distribution over actions depending on the observed state $s \in S$. The optimal policy is the one yielding the largest *return*, that is the expected discounted sum of rewards collected from the initial state following the policy. Tasks specifications based on temporal logic or reward machines naturally induce non-Markovian reward functions. This more expressive framework is commonly referred to as a Non-Markovian-Reward Decision Process (NMRDP). Formally an NMRDP is a tuple $\langle S, A, \bar{R}, P, \gamma \rangle$ equivalent to an MDP except for the non-Markovian reward function $\bar{R} : (S \times A)^* \to \mathbb{R}$, which depends on the entire history of state-action pairs [Bacchus et al., 1996]. Classical reinforcement learning methods cannot be applied directly to NMRDPs.

Linear temporal logic over finite traces. The LTLf logic [De Giacomo and Vardi, 2013] is a variant of LTL [Pnueli, 1977b] defined over finite traces. Logic formulas specified in LTLf are evaluated from propositional Boolean symbols defined in the alphabet $\mathcal{P}$. A finite trace $\sigma = \langle \sigma_0, \sigma_1, \ldots \sigma_n \rangle$ is a sequence of *states* (not to be confused with the states of the MDP). Each state $\sigma_i \subseteq \mathcal{P}$ is a set of propositional symbols true at the i-th timestep, where the finite integer $n \in \mathbb{N}$ specifies the last time step. In LTLf, a formula φ is defined by the grammar $\varphi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \varphi_1 \mathcal{U} \varphi_2$, where $p \in \mathcal{P}$, X is the *next* operator and $\mathcal{U}$ is the *until* operator. Other operators can be defined via standard abbreviations, for instance $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, the *weak next* operator $\tilde{X}\varphi := \neg X \neg\varphi$ and the *eventually* operator $F\varphi := \top \mathcal{U} \varphi$. A formula φ is evaluated over trace σ at a time step t, which is formally denoted as $(\sigma, t) \models \varphi$. We adopt the standard semantic of LTLf, defined by induction on the formula [De Giacomo and Vardi, 2013]. The key aspects on top of the semantic of propositional logic are: $(\sigma, t) \models X\varphi$ if and only if $(\sigma, t + 1) \models \varphi$, and $(\sigma, t) \models \varphi_1 \mathcal{U} \varphi_2$ if and only if $(\sigma, j) \models \varphi_2$ for some $j \geq t$ and $(\sigma, k) \models \varphi_1$ for all $t \leq k < j$. In reward specification based on temporal logic, we interpret propositional symbols in $\mathcal{P}$ as world features, which the agent receive at each time step t. At each time step t, the (binary) reward obtained by the agent is decided by evaluating φ over the current logical state σ_t. Typically, the propositional symbols or fluents in $\mathcal{P}$ represent truth values about the MDP state. For example, De Giacomo et al. [2019] refer to this encoding as a *feature* function $f_i : W \to D_i$ mapping a world state w from the set of world states W to a finite domain D such as $\{\text{True}, \text{False}\}$. A state of the trace $\sigma_t = \langle f_1(w_t), f_2(w_t) \ldots \rangle$ is obtained by extracting the features f_i from the current world state w_t at time t. The clear non-Markovian nature of this reward signal makes the decision problem a NMRDP. A widely adopted solution [e.g., De Giacomo et al., 2020b] is to convert φ into a finite automaton $\mathcal{A}_\varphi = \langle Q, 2^{\mathcal{P}}, \delta, q_0, F \rangle$, where Q is a set of automaton states, $2^{\mathcal{P}}$ is the input alphabet, δ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ the set of accepting (terminal) states. The automaton accepts a trace $\sigma = \langle \sigma_0, \sigma_1 \rangle$ if, starting from q_0 and sequentially consuming inputs from σ, it reaches an accepting state $q \in F$. By combining the NMRDP $\mathcal{M}$ with the automaton $\mathcal{A}_\varphi$, we construct a product MDP $\mathcal{M}' := \langle S', A', R', P', \gamma \rangle$ where the state space $S' = S \times Q$ tracks both the environment state and the automaton state. Transition and rewards in the product MDP are Markovian, as the automaton state capture the necessary information about the history. Hence, $\mathcal{M}'$ is equivalent to the original NMRDP, but can be solved using standard RL techniques.

4 Reward Specification via LTLf$^{\text{MT}}$

LTLf$^{\text{MT}}$ [Geatti et al., 2022] extends the expressiveness of temporal logic by allowing predicates in $\mathcal{P}$ to be first-order formulas over arbitrary theories. A predicate $p \in \mathcal{P}$ is no longer a Boolean symbol but can instead represent a first-order formula, interpreted according to a chosen background theory. In this section, we formally define the syntax of LTLf$^{\text{MT}}$ showing how different theories are integrated into this framework. Then we propose a propositionalizable *fragment* of LTLf$^{\text{MT}}$ retaining sufficient expressiveness for reward specification in RL. Finally, we show how to translate a formula of this fragment into a Deterministic Finite Automaton (DFA) that can be used to define a product MDP for reinforcement learning.

4.1 Linear Temporal Logic Modulo Theories with finite traces

To represent predicates as first order formulas, in LTLf$^{\text{MT}}$, the alphabet $\mathcal{P}$ is replaced by a multi-sorted first-order signature $\Sigma = \mathcal{S} \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{F} \cup \mathcal{V} \cup \mathcal{W}$ where, $\mathcal{S}$ is a set of sort symbols (i.e., data types), $\mathcal{C}$ is a set of constants, $\mathcal{P}$ is the set of predicates,[1] $\mathcal{F}$ the set of functions, $\mathcal{V}$ and $\mathcal{W}$ are the sets of variables and quantified variables respectively. A Σ-term t serves as a building block for constructing atomic formulas in LTLf$^{\text{MT}}$, and is generated by the following grammar:

$$t := v \mid w \mid c \mid f(t_1, \ldots, t_k) \mid \bigcirc v \mid \circledS v \tag{1}$$

where $c \in \mathcal{C}$, $v \in \mathcal{V}$, $w \in \mathcal{W}$ and $f \in \mathcal{F}$ is a function symbol of arity k. The operators $\bigcirc$ and $\circledS$ denote the lookahead and weak lookahead operators, respectively, similar to the next X and weak next $\tilde{X}$ in LTLf, but acting on Σ-variables; they are used to relate the value of variables at a given time to their value at the following instant. The complete grammar of LTLf$^{\text{MT}}$ is defined as:

$$\begin{aligned} &\alpha := p(t_1, \ldots, t_k), \\ &\lambda := \alpha \mid \neg\alpha \mid \lambda_1 \vee \lambda_2 \mid \lambda_1 \wedge \lambda_2 \mid \exists v \lambda \mid \forall v \lambda, \\ &\varphi := \top \mid \lambda \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \tilde{X}\varphi \mid \varphi_1 \mathcal{U} \varphi_2. \end{aligned} \tag{2}$$

Compared to LTLf, this grammar introduces two additional layers α and λ. Intuitively, α defines atomic predicates over terms, and λ allows first-order combinations over these atoms. The top-layer formulas φ are standard temporal formulas over these first-order expressions, similar to LTLf. Theories are integrated into this framework via the interpretation of Σ-terms. A theory $\mathcal{T}$ can be defined as a set of Σ-structures, i.e., sets that interpret all the symbols in Σ (so, also Σ-terms) which are called models of the theory. Intuitively, a Σ-signature defines the available symbols (e.g., $1, 2, +, >, \ldots$), a Σ-structure provides a concrete interpretation (e.g., $+(1, 2)$ is interpreted as integer addition). A theory $\mathcal{T}$ provides interpretations for each symbol in Σ. For example, the Linear Integer Arithmetic (LIA) theory introduces a *sort* for `Integer` data types into $\mathcal{S}$, arithmetic operations such as $+$ and its inverse $-$ in the set $\mathcal{F}$, constants and variables valued in $\mathbb{Z}$ via $\mathcal{C}$, $\mathcal{V}$ and $\mathcal{W}$, and comparison predicates (e.g., $=$, $<$) into $\mathcal{P}$. Multiple theories can be combined, including their Σ-signatures, and their interpretations are given through combined Σ-structures. Formally a $\mathcal{T}$-state $s = (M, \mu)$, contains two elements, $M \in \mathcal{T}$ (i.e., a $\mathcal{T}$-model) is a first-order Σ-structure that interprets all the symbols in Σ and $\mu : \mathcal{V} \to \text{dom}(M)$ a function mapping non-quantified variables to their domain of reference. Quantified variables are interpreted by a separate environment function $\xi : \mathcal{W} \to \text{dom}(M)$. A trace (or word), in LTLf$^{\text{MT}}$ is a finite sequence of $\mathcal{T}$-state over time-steps, $\sigma = \langle (M, \mu_0), \ldots, (M, \mu_n) \rangle$ where $M \in \mathcal{T}$ is fixed across the trace, and μ_i may differ, reflecting the evolution of variables over time. The evaluation of a term t at time i in a trace σ under variable environment ξ is defined using the functions μ_i and ξ in the base case of variables in $\mathcal{V}$ and $\mathcal{W}$, resp., and defined inductively in the usual way for constants and functional terms. The satisfaction relation for an LTLf$^{\text{MT}}$ formula φ over a trace σ, at time-step i with environment function ξ is denoted as $\sigma, \xi, i \models \varphi$ ad follows the usual definition of first-order semantics. This satisfaction relation is used

[1]Note that in LTLf$^{\text{MT}}$ there is no alphabet. Hence, from now on $\mathcal{P}$ denotes the predicate symbols of the first-order signature Σ.

to define the *satisfaction modulo $\mathcal{T}$* of an LTLf$^{\text{MT}}$ formula φ over the trace σ at time point time-step i, denoted as $\sigma, i \models \varphi$ and inductively defined in the standard way to deal with the temporal operators (see [Geatti et al., 2022] for the formal definition of the semantics).

4.2 Fragment for Reward Specification in RL

While LTLf$^{\text{MT}}$ framework allows for very general expressions, it is, in general, undecidable [Geatti et al., 2022]. However, it is possible to define a fragment of LTLf$^{\text{MT}}$ that is semi-decidable . Formally, for decidable underlying first-order theories, LTLf$^{\text{MT}}$ has been proven to be semi-decidable (i.e., a positive answer can always be obtained for satisfiable instances, while reasoning might not terminate over unsatisfiable instances), thanks to the semi-decision procedure shown by Geatti et al. [2022], and also implemented in the BLACK solver. [2] Intuitively, for the identifying a decidable fragment of LTLf$^{\text{MT}}$, it a necessary requirement that the underlying theory $\mathcal{T}$ is decidable, but this is not sufficient in general: decidability in first-order extensions of LTLf may still be broken by the interaction of temporal operators and quantifiers, i.e., in the case of LTLf$^{\text{MT}}$, by the presence of the lookaheads operators. We propose a tractable fragment of LTLf$^{\text{MT}}$ for reward specification in reinforcement learning by removing the (weak) lookahead operator $\bigcirc$ ($\circledsim$) from the Σ-layer. While the (weak) next operator X ($\tilde{X}$) in the temporal logic layer is essential for specifying temporal constraints between “subtasks” that the agent may have to complete, the addition of (weak) lookahead operator on the Σ-term, $\bigcirc$ ($\circledsim$), which applies to variables, is neither natural nor beneficial for reward specification. The Σ-layer can be regarded as a data collection interface for capturing unstructured and heterogenous environmental information. We argue that defining temporal constraints on data collection is not useful for reward specification, and that the Σ-layer should just be used to capture immediate information available to the agent. Note that our fragment of LTLf$^{\text{MT}}$ still supports arbitrary first-order theories.

$$\begin{aligned}
t &:= v \mid w \mid c \mid f(t_1, \ldots, t_k) \\
\alpha &:= p(t_1, \ldots, t_k) \\
\lambda &:= \alpha \mid \neg\alpha \mid \lambda_1 \vee \lambda_2 \mid \lambda_1 \wedge \lambda_2 \mid \exists v \lambda \mid \forall v \lambda \\
\varphi &:= \top \mid \lambda \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \tilde{X}\varphi \mid \varphi_1 \mathcal{U} \varphi_2
\end{aligned} \tag{3}$$

In this fragment, the source of undecidability becomes the theory itself and its interaction with quantifiers. Assuming the theory itself is decidable, we can leverage any tool implementing a decision procedure for the theory to deal with the first-order dimension: this is the case if the quantifier-free fragment of the theory is decidable and the quantifiers do not affect decidability, because, for instance, a quantifier elimination procedure is available (as in the case of real arithmetic [Tarski, 1951]). In such scenarios, we can employ a state-of-the-art QE module for eliminating the quantifiers [Collins, 1974], and then universal tools such as SMT solvers [Barrett et al., 2021] to evaluate the obtained quantifier-free formula. Depending on the theory, in cases where the use of quantifiers leads to undecidability, we can further restrict our fragment by removing them. Doing so, we obtain a LTLf$^{\text{MT}}$ fragment similar to “data-LTLf” by Gianola et al. [2024].

4.3 Translation to DFA and product MDP

Geatti et al. [2022] give a non-constructive proof that a Boolean abstraction of any language definable in lookahead-free LTLf$^{\text{MT}}$ can also be defined in LTLf, and vice versa. We take a more practical viewpoint and provide a propositionalization of our fragment in the following sense: (i) a syntactic transformation of lookaead-free LTLf$^{\text{MT}}$ formulas into LTLf formulas, and (ii) a transformation of LTLf$^{\text{MT}}$ traces into LTLf traces. This is enough to define an automaton and a product MDP that can be simulated efficiently using only a theory solver for the elected theory (such as an SMT solver if the theory quantifier-free). Refer to the grammar of our fragment in Equation (3). Given a formula φ from our language, we first substitute each α containing only free variables from $\mathcal{V}$ with a distinct propositional letter. Then, we substitute each remaining λ with a distinct propositional letter. The result is a LTLf formula φ'. Let $\mathcal{P}$ now denote the set of all the employed propositional letters—our new alphabet. A LTLf$^{\text{MT}}$ trace $\sigma = (s_0, \ldots, s_t)$ is mapped to an LTLf trace $\sigma' = (\sigma'_1, \ldots, \sigma'_n)$ by assigning a truth value to each propositional letter in $\mathcal{P}$ at each step. This is done, e.g., with a theorem

[2]https://www.black-sat.org/en/stable/

prover [Kovács and Voronkov, 2013] or a decision procedure for the theory of choice (or an SMT solver if the theory is quantifier-free, as detailed in Section 4.2). Crucially, the problem given to the solver at a given timestep is purely first-order, without any temporal operator. For example, assuming the theory is real arithmetic, if "$\forall y \exists z(y * z > 0 \implies z < y)$" from ϕ was replaced with letter P in ϕ', the solver decides $P \in \sigma'_t$ if and only if $s_t \models \forall y \exists z(y * z > 0 \implies z < y)$, applying efficient quantifier elimination first. At runtime, the propositional abstraction φ' is evaluated on the $LTLf$ trace σ'. The SMT solver plays the role of an extremely general *labeling function* (cf. Section 2). For an RL-friendly implementation, we can translate φ' into a Deterministic Finite Automaton (DFA) $\mathcal{A}_{\varphi'} = \langle Q, 2^{\mathcal{P}}, \delta, q_0, F \rangle$ and use it to define a product MDP $M' := \langle S', A', R', P', \gamma \rangle$ with extended state space $S' = S \times Q$, exactly like in LTLf (cf. Section 3). We can do this automatically using the translation technique defined in Zhu et al. [2019][3] At training time, state transitions in the DFA are resolved by the SMT solver. This is a special kind of reward machine (specifically, a *simple* reward machine with a Boolean-valued reward function, cf. [Icarte et al., 2018]), again with the SMT solver playing the role of the labeling function.

5 Reinforcement Learning with LTLf$^{\text{MT}}$ Rewards

In this section, we provide an RL solution to the tasks defined using our proposed specification language, with a focus on the continuous control-domain.

Let us first recap our reward specification pipeline, stressing that it reduces significantly manual interventions compared to other methods. The end goal of the specification is to construct a product MDP to which standard RL algorithms (and more sophisticated techniques inspired by reward machines) can be applied. First, depending on the domain of interest (e.g., continuous control), we pick a theory (e.g., NRA) with an associated SMT solver (with all the precautions of Section 4.2). Several different tasks can be specified within the domain of interest. From the user defining the task, or *reward engineer*, the only input we require is a formula φ written in the lookahead-free LTLf$^{\text{MT}}$ fragment defined in Section 4.2. Compared to existing reward specification frameworks (cf. Section 2), the user need not manually implement a labeling function or anything similar. At most, to enable *multi-goal RL* (for example, to use HER as we will detail later) the user needs to provide a dictionary assignment of first-order constants appearing in the formula to numerical values. In this way, the task is "parametrized" by (a subset of) the constants. In the following, we will focus on continuous control, both to provide an intuitive, realistic example of our reward specification pipeline and study the effectiveness of HER in dealing with sparse rewards generated by logic specifications.

Continuous Control Case Study Common tasks in continuous control consist in controlling a variable to a target value or range, or a vector of variables to a target point. For example, a target position in space for a robot, a desired temperature for a HVAC system, a speed for cruise control. These "subtasks" can be described in terms of L^p distances between state vectors (e.g., Euclidean distance, box contraints) This can be done using real arithmetic, and subtasks can be combined into more complex tasks using temporal operators. Hence, for continuous control, we can select Non-Linear Real Arithmetic (NRA) as our theory for lookahead-free LTLf$^{\text{MT}}$, with all the computational advantages discussed in Section 4.2. For example, consider an autonomous car moving in 2-dimensional space within a parking area. A state $s \in \mathbb{R}^2$ represents spatial coordinates. We specify a (parametric) task where the agent must reach a point A in space first, then a point B, without ever going through a certain "unsafe" area. The task can be specified in our lookahead-free LTLf$^{\text{MT}}$ fragment as $\varphi = \neg F \neg (x \geq x_{min} \land x \leq x_{max}) \land F((x - x_a)^2 + (y - y_a)^2 < r_a^2 \land F((x - x_b)^2 + (y - y_b)^2 < r_b^2))$. In this case, we use an SMT solver suited for Linear Real Arithmetic (LRA) and Non-Linear Real Arithmetic (NRA) to propositionalize φ and obtain $\varphi' := \neg F \neg \alpha_1 \land F(\alpha_2 \land F(\alpha_3)$.

We use this example to stress the advantages of our reward specification framework. First, the input we require to the user to specify the task is very simple. It consists of a simple string representing the specification formula φ, along with an optional assignment of constants to numerical values to parametrize the task.[4]. In contrast, existing approaches often require complex inputs. For example, reward machines as proposed by [Icarte et al., 2018] require explicit specification of the automaton.

[3]In our implementation, we use the `LTLf2DFA` library (`https://github.com/whitemech/LTLf2DFA`)

[4]This can be as simple as a `json` configuration file specifying the names of the constants and the values

They also require an implementation of the labeling function, just as several logic-based approaches discussed in 2. Moreover, defining new tasks for the same domain can be very quick in our framework. In the continuous control domain we have outlined, NRA allows defining many new tasks by *just* writing new formulas. Even more substantial modifications to the domain just require changing the theory (for example, from NRA to linear arithmetic), which could be almost effortless depending on the availability of efficient solvers. Finally, in the case of continuous control, our framework makes it easy to handle sparse rewards with HER, as shown in the next section.

5.1 Addressing Sparsity in the Product MDP

A well known issue of logic-based reward specification methods is the sparsity of rewards. For the continuous-control domain, we propose to combine the idea of counterfactual reasoning (CRM) from reward machines [Icarte et al., 2022] with Hindsight Experience Replay (HER). Both techniques work at the data collection level, generating artificial experiences to improve learning. Both can be applied to trajectories generated from the product MDP $\tau = \{< s_0, q_0 >, a_0, r_1, < s_1, q_1 >, \ldots, < s_T, q_T >\}$. Counterfactual reasoning exploits the underlying reward automaton $\mathcal{A}_\varphi$, producing additional "fake" experience by replacing the DFA state with other possible ones. This is always possible since the DFA, differently from the MDP, is simulated by the agent itself. Namely, given a single experience $\{\langle s_t, q_t \rangle, a_t, r_{t+1}, \langle s_{t+1}, q_{t+1} \rangle\}$, we can create $|Q|$ artificial transitions replacing q_t with any other $q \in Q$ and recomputing the reward. HER is designed for goal-based scenarios with sparse binary reward feedback. It improves sample efficiency by learning from unsuccessful trajectories, exploiting the state-generalization capabilities of neural networks. HER requires the definition of a "goal-based problem" consisting of a goal space $\mathcal{G}$, a predicate reward function $f_g : S \to \{0, 1\}$ that checks whether goal $g \in \mathcal{G}$ is achieved in state $s \in S$, and a mapping $m : S \to \mathcal{G}$ s.t. $\forall s \in S, f_{m(s)}(s) = 1$ used to generate artificial goals $g' = m(s)$. A goal-based transition is a tuple $((s_t, g), a_t, r_{t+1}, (s_{t+1}, g))$. First, we show how to adapt CRM and HER to our framework, then we combine them to exploit their synergyto mitigate reward sparsity.

Adapting HER to the product MDP. HER requires a goal specification, which is not intuitive to define over the states of the product MDP and is usually provided by the user. To automate HER, we define as "true" goal of the product MDP the state $\langle s, q \rangle \in S'$, where q is the predecessor of an accepting state $q_f \in F$ on the automaton $\mathcal{A}_{\varphi'}$ and s is such that $q_f = \delta_q(q, L(s, \cdot, \cdot))$, that is a state triggering the transition to q_f. Then, the goal-based specification can be defined as follows: $\mathcal{G} = S'$, $f_g(s) = \mathbb{I}\left[m(s) = g\right]$, and $m(s) = s$, where $s \in S'$ is a state of the product MDP.

Combining CRM and HER. Given a trajectory τ, we add to the replay buffer $|Q|$ artificial experiences obtained by copying τ and applying counterfactual reasoning as described above. Then, we apply HER to each (real or counterfactual) trajectory generating a total of $2|Q|$ artificial experiences, all added to the replay buffer. The intermediate goals of HER are defined using the last state visited in each trajectory τ [Andrychowicz et al., 2017]. In the next section we compare both the separate and combined approaches.

6 Experiments

In this section we use Deep Deterministic Policy Gradients (DDPG) [Lillicrap et al., 2016] to learn an policy from the product MDP constructed for the *parking environment*. We use as baselines different approaches, operating at the *replay buffer* level, presented in Section 5.1. **Baseline**: the task is learned directly from the product MDP using plain DDPG. **CRM**: CRM is applied to the DFA that specifies the task generating virtual experiences for DDPG. **HER**: HER is applied directly to the augmented state of the product MDP, without using CRM. **CRM-HER**: our main proposed approach, where HER is applied to real experiences and counterfactual experiences generated with CRM. Our goal is to assess how HER and CRM-HER compare to prior methods across tasks of varying complexity, and whether the supposed synergy of CRM and HER translates into practical advantages.

Environments and tasks. We conducted experiments on a range of tasks (i.e., different φ formulae) defined within two distinct continuous control environments: *Parking* and *Reacher*. The *Parking* environment, from the *HighwayEnv* collection [Leurent, 2018], involves an agent controlling a car in an empty parking lot. We modified this environment to accommodate task specification defined by

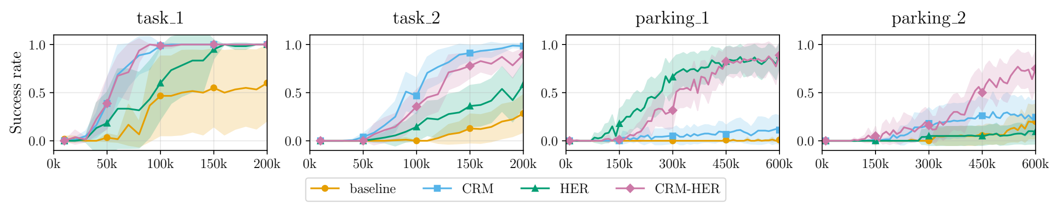

Figure 1: Success rate of each baselines for parking environment in 4 different tasks, with 95% bootstrap confidence interval over 20 independent experiments. The x-axis reports the training steps of DDPG. Task names are specified above each plot and are ordered left-to-right by task complexity.

arbitrary LTLf^{MT}-NRA formulas. In each of the two underlying environments, we defined multiple tasks of increasing complexity using our specification pipeline detailed in Section 5. The complexity of these tasks stems from two main factors: the size and structure of the formula used to define the task (reflected in the complexity of the resulting DFA/product MDP) and the difficulty of discovering the goal, which becomes particularly challenging due to the presence of sparse rewards. Complete details about task specification (including LTLf^{MT} formulas) and results for *Reacher* are in appendix.

Analysis of results. Figure 1 presents the results obtained by averaging performance over multiple training runs for each method. Performance is measured as the *success rate* during evaluation episodes conducted periodically throughout training. An episode is considered successful if the agent reaches the goal within the episode's time limit. Our results show that, in most tasks, the baseline method struggles to consistently learn a policy capable of reaching the objective, lagging behind the other approaches. This trend holds across all difficulty levels but becomes especially pronounced in the more complex tasks, where it completely fails to discover a successful policy (e.g., see results for tasks `parking_1`, `parking_2`). In most cases, CRM is more effective at handling the increase in complexity of the DFA. This is particularly evident in `tasks_1` and `task_2` of the Parking environment , where its performance is on par with the best-performing methods. However, when the goal is difficult to discover, CRM alone offers little benefit, and its performance falls behind that of HER-based approaches (see especially `parking_1`). RM-HER exhibits mixed performance, with results varying depending on the nature of the task. In tasks where the primary challenge lies in goal discovery, it performs well, clearly demonstrating the benefits of HER (e.g., see `task_2`, `parking_2`). In other tasks, however, the improvement over the baseline is less pronounced or, in some cases, barely noticeable. CRM-HER consistently performs well across all tasks, either achieving the best results or coming close to the top-performing method. The results indicate that CRM-HER effectively combines the strengths of both CRM and HER, enabling it to handle increasing task complexity. The performance advantage becomes especially clear in the more challenging tasks—most notably in `parking_2` task, where it is the only method able to learn a successful policy.

7 Conclusions

We proposed a new framework for reward specification in RL based on a fragment of the LTLf^{MT} logic. Compared to LTLf, our language is more expressive as it allows predicates to be first-order logic formulas over arbitrary theories. Although the definition of the logic-augmented MDP follows the standard procedure of LTLf, the increased expressiveness of the language has two important consequences for the user responsible of specifying the reward: (i) the user is relieved of the burden of *coding* labeling functions, which are replaced by off-the shelf SMT solvers, only having to *write a logical formula* and (ii) in continuous-control domains, with minimal effort, they can provide a goal specification for HER using the same variables used in the formula. We exploit this second advantage to address the inevitable sparsity of the rewards resulting from such a logic-based specification, combining the CRM technique of reward machines with HER. Experiments in benchmark continous-control domains on tasks specified with lookahead-free LTLf-modulo-NRA show promising results. Future work should explore integration of these techniques with hierarchical RL methods [Icarte et al., 2022]. Finally, our reincorporation of first-order terms *inside* the specification language paves the way to future further combinations of symbolic aspects with *continous-space* RL, such as the specification of safety constraints, of continuous rewards, of multi-task problems, ultimately to the formal verification of learning processes in continuous domains.

Acknowledgments

Funded by the European Union – Next Generation EU within the project NRPP M4C2, Investment 1.3 DD. 341 – 15 March 2022 – FAIR – Future Artificial Intelligence Research – Spoke 4 – PE00000013 – D53C22002380006.

This work was partially supported by the ‘OptiGov’ project, with ref. n. 2024.07385.IACDC (DOI: 10.54499/2024.07385.IACDC), fully funded by the ‘Plano de Recuperação e Resiliência’ (PRR) under the investment ‘RE-C05-i08 - Ciência Mais Digital’ (measure ‘RE-C05-i08.m04’), framed within the financing agreement signed between the ‘Estrutura de Missão Recuperar Portugal’ (EMRP) and Fundação para a Ciência e a Tecnologia, I.P. (FCT) as an intermediary beneficiary.

References

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NIPS*, pages 5048–5058, 2017.

Brandon Araki, Xiao Li, Kiran Vodrahalli, Jonathan A. DeCastro, Micah J. Fry, and Daniela Rus. The logical options framework. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 307–317. PMLR, 2021.

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

Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In Dana Fisman and Grigore Rosu, editors, *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*, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022. doi: 10.1007/978-3-030-99524-9_24. URL https://doi.org/10.1007/978-3-030-99524-9_24.

Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1267–1329. IOS Press, 2021. doi: 10.3233/FAIA201017. URL https://doi.org/10.3233/FAIA201017.

Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *ICRA*, pages 10349–10355. IEEE, 2020.

Ronen I. Brafman, Giuseppe De Giacomo, and Fabio Patrizi. LTLf/LDLf non-markovian rewards. In *AAAI*, pages 1771–1778. AAAI Press, 2018.

Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robotics Autom. Lett.*, 6(4):7973–7980, 2021.

George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition-preliminary report. *SIGSAM Bull.*, 8(3):80–90, 1974. doi: 10.1145/1086837.1086852. URL https://doi.org/10.1145/1086837.1086852.

Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Ijcai*, volume 13, pages 854–860, 2013.

Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *Proceedings of the international conference on automated planning and scheduling*, volume 29, pages 128–136, 2019.

Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi, and Alessandro Ronca. Temporal logic monitoring rewards via transducers. In *KR*, pages 860–870, 2020a.

Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Restraining bolts for reinforcement learning agents. In *AAAI*, pages 13659–13662. AAAI Press, 2020b.

Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008. doi: 10.1007/978-3-540-78800-3_24. URL https://doi.org/10.1007/978-3-540-78800-3_24.

Marco Dorigo and Marco Colombetti. Robot shaping: Developing autonomous agents through learning. *Artif. Intell.*, 71(2):321–370, 1994.

Marco Faella and Gennaro Parlato. A unified automata-theoretic approach to LTL_f modulo theories. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz, editors, *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 1254–1261. IOS Press, 2024. doi: 10.3233/FAIA240622. URL https://doi.org/10.3233/FAIA240622.

Luca Geatti, Alessandro Gianola, and Nicola Gigante. Linear temporal logic modulo theories over finite traces. In *IJCAI*, volume 22, pages 2641–2647, 2022.

Luca Geatti, Alessandro Gianola, Nicola Gigante, and Sarah Winkler. Decidable fragments of LTL_f modulo theories. In Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Radulescu, editors, *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 811–818. IOS Press, 2023. doi: 10.3233/FAIA230348. URL https://doi.org/10.3233/FAIA230348.

Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Safety verification and universal invariants for relational action bases. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 3248–3257. ijcai.org, 2023. doi: 10.24963/IJCAI.2023/362. URL https://doi.org/10.24963/ijcai.2023/362.

Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTLf and LDLf goals. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4729–4735. ijcai.org, 2018. doi: 10.24963/IJCAI.2018/657. URL https://doi.org/10.24963/ijcai.2018/657.

Alessandro Gianola. *Verification of Data-Aware Processes via Satisfiability Modulo Theories*, volume 470 of *Lecture Notes in Business Information Processing*. Springer, 2023. doi: 10.1007/978-3-031-42746-6. URL https://doi.org/10.1007/978-3-031-42746-6.

Alessandro Gianola, Marco Montali, and Sarah Winkler. Linear-time verification of data-aware processes modulo theories via covers and automata. In *AAAI*, pages 10525–10534. AAAI Press, 2024.

Charles Gretton. A more expressive behavioral logic for decision-theoretic planning. In *PRICAI*, volume 8862 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2014.

Charles Gretton, David Price, and Sylvie Thiébaux. Implementation and comparison of solution methods for decision processes with non-markovian rewards. In *UAI*, pages 289–296. Morgan Kaufmann, 2003.

Charles Gretton, Froduald Kabanza, David Price, John K. Slaney, and Sylvie Thiébaux. Decision-theoretic planning with non-markovian rewards. *CoRR*, abs/1109.2355, 2011.

Hosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Certified reinforcement learning with logic guidance. *Artif. Intell.*, 322:103949, 2023.

Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *FORMATS*, volume 12288 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2020.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

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, 2022.

Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. Temporal-logic-based reward shaping for continuing reinforcement learning tasks. In *AAAI*, pages 7995–8003. AAAI Press, 2021.

Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. In *NeurIPS*, pages 13021–13030, 2019.

Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013. doi: 10.1007/978-3-642-39799-8_1. URL `https://doi.org/10.1007/978-3-642-39799-8_1`.

Bruno Lacerda, David Parker, and Nick Hawes. Optimal policy generation for partially satisfiable co-safe LTL specifications. In *IJCAI*, pages 1587–1593. AAAI Press, 2015.

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

Andrew C. Li, Zizhao Chen, Toryn Q. Klassen, Pashootan Vaezipoor, Rodrigo Toro Icarte, and Sheila A. McIlraith. Reward machines for deep RL in noisy and uncertain environments. In *NeurIPS*, 2024.

Xiao Li, Cristian Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *IROS*, pages 3834–3839. IEEE, 2017.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, pages 278–287. Morgan Kaufmann, 1999.

Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57. IEEE Computer Society, 1977a. doi: 10.1109/SFCS.1977.32. URL `https://doi.org/10.1109/SFCS.1977.32`.

Amir Pnueli. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, pages 46–57. ieee, 1977b.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Andoni Rodríguez and César Sánchez. Boolean abstractions for realizability modulo theories. In Constantin Enea and Akash Lal, editors, *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part III*, volume 13966 of *Lecture Notes in Computer Science*, pages 305–328. Springer, 2023. doi: 10.1007/978-3-031-37709-9_15. URL `https://doi.org/10.1007/978-3-031-37709-9_15`.

Andoni Rodríguez and César Sánchez. Adaptive reactive synthesis for LTL and LTLf modulo theories. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 10679–10686. AAAI Press, 2024. doi: 10.1609/AAAI.V38I9.28939. URL `https://doi.org/10.1609/aaai.v38i9.28939`.

Andoni Rodríguez, Guy Amir, Davide Corsi, César Sánchez, and Guy Katz. Shield synthesis for LTL modulo theories. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 15134–15142. AAAI Press, 2025. doi: 10.1609/AAAI.V39I14.33660. URL `https://doi.org/10.1609/aaai.v39i14.33660`.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Alfred Tarski. A decision method for elementary algebra and geometry. *University of California Press*, 1951.

Sylvie Thiébaux, Froduald Kabanza, and John K. Slaney. Anytime state-based solution methods for decision processes with non-markovian rewards. In *UAI*, pages 501–510. Morgan Kaufmann, 2002.

Shufang Zhu, Geguang Pu, and Moshe Y. Vardi. First-order vs. second-order encodings for LTLf-to-automata translation, 2019. URL `https://arxiv.org/abs/1901.06108`.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: claims on the expressiveness of the new reward specification framework are substantiated in Section 4. Claims on the effectiveness of HER combined with CRM are substantiated empirically in Section 6.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: the practical scope of the proposed framework and potential computational barriers are discussed in Section 4.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: details to reproduce the experiments are provided in appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] ,

   Justification: experimental details are provided in appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: learning curves show mean and 95% confidence intervals over independent experiments, as stated in captions.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: details on compute resources are provided in appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: the authors reviewed the code of ethics guideline and did not identify any issues.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA] .

    Justification: the authors do not foresee any specific societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA] .

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA] .

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] .

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA] .

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Task Descriptions

In this section, we provide additional details about the experimental setup. We describe the environments we used and the tasks (both in formula and text description) we defined for our experiments. Assignments of numerical values to certain constants display our manual configuration of the tasks (goal assignment), but also which constants HER can manipulate to define fictitious goals.

A.1 Parking Environment

The parking environment [Leurent, 2018] represents a two-dimensional parking lot where the agent controls a car, represented by a continuous vector $s \in \mathbb{R}^6$. The state contains information about spatial coordinates x, y, velocities v_x, v_y, and orientation $\alpha = \sin\theta, \beta = \cos\theta$ where θ is the angle of yaw. The action space is continuous and 2-dimensional (throttle and steering angle). In the following, we list the tasks we defined for this environment. The only variables appearing in the formulas are x, y, α, β, while a, b, c, d are constants.

Task 1 In `task_1` the agent is required to first navigate a checkpoint position $A = (-0.2, -0.08)$, then to move toward a goal position $G = (a, b)$ in the parking lot, respecting this temporal order. The formula describing this task consists of a distance constraint on the spatial coordinate variables of the car (x, y), with a tolerance of $r = 0.03$:

$$\varphi_1 = \; F(\; (x+0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge XF\; (x-a)^2 + (y-b)^2 < 0.03^2 \;).$$

Goal assignment: $\{a = 0.2, b = 0.08\}$.

Task 2 In `task_2`, we introduce an additional target position $B = (e, f)$. The agent is required to navigate both checkpoints A, B in any order and then move to the goal G. Below the corresponding formula:

$$\begin{aligned}\varphi_2 = F(\; &(x+0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge X\\ &F(\; (x-0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge XF\; (x-a)^2 + (y-b)^2 < 0.03^2 \;)\;)\\ &\vee\\ &F(\; (x-0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge X\\ &F(\; (x+0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge XF\; (x-a)^2 + (y-b)^2 < 0.03^2 \;)\;).\end{aligned}$$

Goal assignment: $\{a = 0.2, b = 0.08\}$.

Task 3 This task, denoted in Figure 1 as `parking_1` introduces a stricter specification for the goal G. After going through checkpoint A, instead of just reaching a position in the parking lot, the agent is required to correctly park the car in a specific spot, which means that the car must also meet orientation constraints. This is formalized via a box constraint (weighted L_1 norm) over spatial and orientation variables x, y, α, β as follows:

$$\begin{aligned}\varphi_3 = &F\;(\;(x+0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge X\\ &\quad F\;(|x-a| + 0.2|y-b| + 0.02|\alpha-c| + 0.02|\beta-d| < 0.0144\;)\;).\end{aligned}$$

where $r = 0.0144$ is the tolerance threshold. Goal assignment: $\{a = 0.18, b = 0.14, c = 0, d = 1\}$.

Task 4 This task referred to as `parking_2` is the most difficult in our tests, as it combines the complexity of previous tasks. It requires the agent to pass through the two checkpoint positions A, B (in any order), then to correctly park the car in a specific spot.

$$\begin{aligned}\varphi_4 = F(\; &(x+0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge XF(\; (x-0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge X\\ &F\;(|x-a| + 0.2|y-b| + 0.02|\alpha-c| + 0.02|\beta-d| < 0.0144\;)\;)\\ &\vee\\ &F(\; (x-0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge XF(\; (x+0.2)^2 + (y+0.08)^2 < 0.03^2 \;\wedge X\\ &F\;(|x-a| + 0.2|y-b| + 0.02|\alpha-c| + 0.02|\beta-d| < 0.0144\;)\;).\end{aligned}$$

Goal assignment: $\{a = 0.18, b = 0.14, c = 0, d = 1\}$.

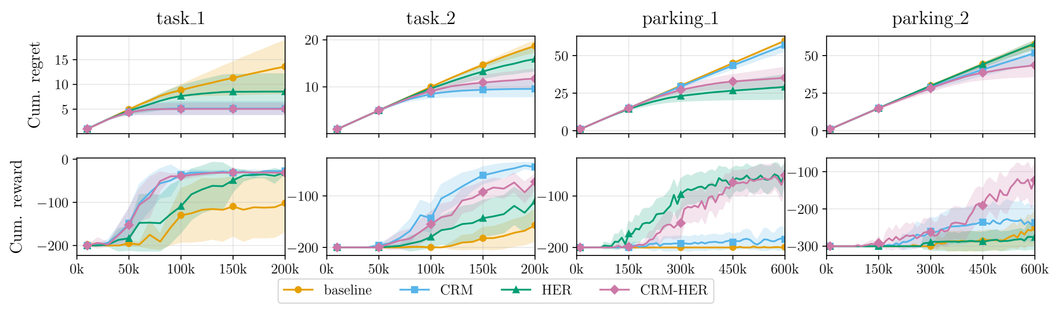

Figure 2: Regret and cumulative reward plot for each task on Parking environment.

Analysis of results for parking Additional results for parking are shown in Figure 3, where for each task we report cumulative regret and cumulative reward over 20 runs. The cumulative regret is measured against an ideal policy that always succeeds. These additional results reflect the analysis made in Section 6.

A.2 Reacher Environment

Reacher is a two-jointed robotic arm moving in a two-dimensional space. We use a variant of `Reacher-v5`[5] with a modified state space. Our state $s \in \mathbb{R}^{11}$ encodes the sine and cosine of the angle of each arm $\alpha_1, \beta_1, \alpha_2, \beta_2$, the angular velocity of each arm ω_1, ω_2, the spatial position of the target (x_t, y_t), and the coordinates of the arm hand $(x, y, z = 0)$. The action space is continuous and 2-dimensional (torques of two hinges). Below we describe the tasks we defined for reacher. The only variables appearing in the formulas are x and y, while a and b are constants.

Task 1 In the task named `task_1` in Figure 3, the agent must move the hand (x, y) to the left, which means moving in any position described by the box constraint $x < -0.19$, then move it to a specified goal position $G = (a, b)$, within $r_g = 0.01$ tolerance radius. The formula is as follows:

$$\varphi_1 = F(\ x \leq -0.19\ \wedge XF\ (x-a)^2 + (y-b)^2 < 0.01^2\ ).$$

Goal assignment: $\{a = 0.1, b = 0.1\}$.

Task 2 Task `task_2` requires an additional step. After moving the arm to the left, the agent has to move the hand close to it's base $S = (0, 0)$, with a tolerance distance of $r_s = 0.03$, only then it must reach the goal position G. The formula is as follows:

$$\varphi_2 = F(\ x \leq -0.19\ \wedge XF\ (\ x^2 + y^2 < 0.03^2\ \wedge XF\ (x-a)^2 + (y-b)^2 < 0.01^2\ )).$$

Goal assignment: $\{a = 0.1, b = 0.1\}$.

Task 3 Task `task_3` is similar to `task_2`, with a stricter tolerance $r_g = 0.005$ to reach the goal position, reducing the tolerance distance from 0.01 to 0.005, which increases the complexity significantly reducing the accepted goal area. The formula, is almost equal to previous task:

$$\varphi_3 = F(\ x \leq -0.19\ \wedge XF\ (\ x^2 + y^2 < 0.03^2\ \wedge XF\ (x-a)^2 + (y-b)^2 < 0.005^2\ ))$$

Goal assignment: $\{a = 0.1, b = 0.1\}$.

Analysis of results for reacher Results for the Reacher environment are shown in Figure 3. As before, we compare success rate, cumulative regret and cumulative reward across 20 runs. While no single technique strictly dominates in terms of success rate or cumulative reward, CRM-HER consistently performs comparably or better than other approaches and demonstrates faster convergence in tasks `task_1` and `task_2`. This improvement becomes even more evident when analyzing cumulative regret, where CRM-HER emerges as a consistently preferable choice over other methods.

[5]`https://gymnasium.farama.org/environments/mujoco/reacher/`

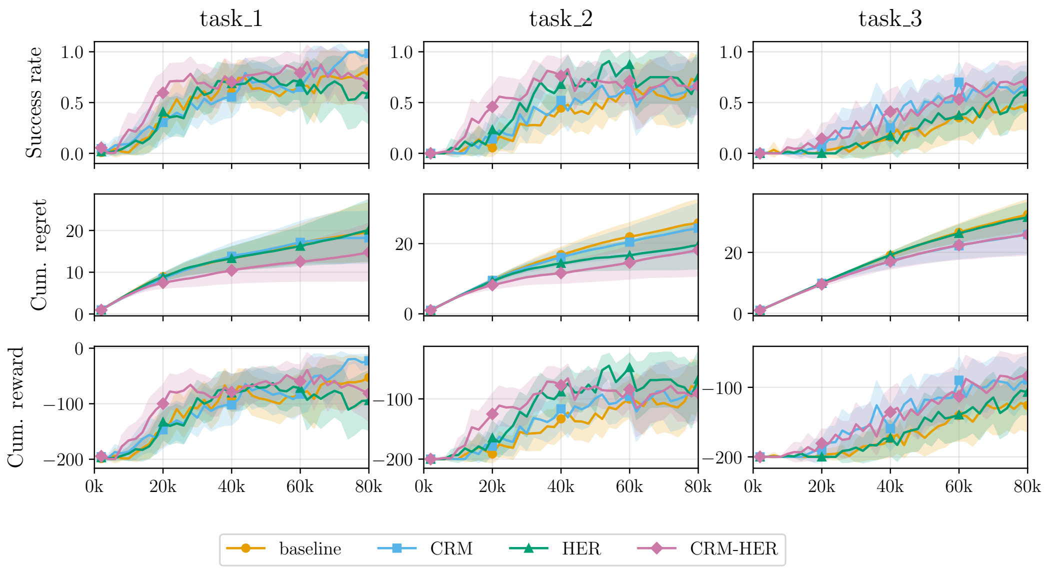

Figure 3: Success rate, cumulative regret and cumulative reward on each task for the reacher environment.

Parking Environment Hyperparameters

	Task 1	**Task 2**	**Task 3**	**Taks 4**
episode length	200	200	200	300
buffer size	10^6	10^6	10^6	10^6
batch size	256	256	256	1024
actor networks	MLP $(400, 300)$	MLP $(400, 300)$	MLP $(400, 300)$	MLP $(400, 300)$
critic networks	MLP $(400, 300)$	MLP $(400, 300)$	MLP $(400, 300)$	MLP $(400, 300)$
α	0.001	0.001	0.001	0.001
τ	0.005	0.005	0.005	0.005
γ	0.99	0.99	0.99	0.99

Table 1: Table showing the hyperparameters used for each task in our experiments in the parking environment.

B Hyperparameters

Tables 1 and 2 show the hyperparameters selected for the learning process in each environment and task. Each parameter is described as follows:

Reacher Environment Hyperparameters

	Task 1	**Task 2**	**Task 3**
episode length	200	200	200
buffer size	10^6	10^6	10^6
batch size	256	256	256
actor networks	MLP $(400, 300)$	MLP $(400, 300)$	MLP $(400, 300)$
critic networks	MLP $(400, 300)$	MLP $(400, 300)$	MLP $(400, 300)$
α	0.001	0.001	0.001
τ	0.005	0.005	0.005
γ	0.99	0.99	0.99

Table 2: Table showing the hyperparameters used for each task in our experiments in the reacher environment.

- "MLP $(400, 300)$" denotes a multilayer perceptron with two hidden layers consisting of 400 and 300 units, respectively.
- α indicates the learning rate used by the ADAM optimizer for gradient updates.
- τ represents the soft update coefficient used in the Polyak averaging of the networks with respect to their corresponding target networks.
- γ is the discount factor.

Most experiments adopt the default hyperparameters provided by the DDPG implementation in *Stable Baselines 3* [Raffin et al., 2021]. The only exception is Task 4 in the parking environment, for which the batch size was increased to allow the algorithms to learn a meaningful policy.

C PC specs

All the experiments were implemented in `Python3.10` and executed on a machine with the following hardware configuration:

- **CPU**: AMD Ryzen 9 7950X
- **GPU**: NVIDIA RTX 4080 Super
- **RAM**: 32GB DDR5 6000MHz