Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An analytical framework for studying subtask difficulty and error accumulation using Reward Machines

Uriel Robles-Cervantes
Universidad de Guanajuato
Mario-Alberto Ibarra-Manzano
Universidad de Guanajuato
Alan Ortega-Gonzalez
Universidad de Guanajuato
Dora-Luz Almanza-Ojeda
dora.almanza@ugto.mx
Universidad de Guanajuato
Fernando Fernández
Universidad Carlos III de Madrid

Research Article

Keywords:

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11861772/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

An analytical framework for studying subtask difficulty and error accumulation using Reward Machines

Uriel Robles-Cervantes[1†], Mario-Alberto Ibarra-Manzano[1†], Alan Ortega-Gonzalez[1†], Dora-Luz Almanza-Ojeda[1*†], Fernando Fernández[2†]

[1]Electronics Engineering Department, DICIS, University of Guanajuato, Carretera Salamanca - Valle de Santiago km 3.5 + 1.8, Comunidad Palo Blanco, Salamanca, 36885, Guanajuato, Mexico.
[2]Universidad Carlos III de Madrid, Avenida de la Universidad 30, Leganés, 28911, Madrid, Spain.

*Corresponding author(s). E-mail(s): dora.almanza@ugto.mx;
Contributing authors: u.roblescervantes@ugto.mx; ibarram@ugto.mx; a.ortegagonzalez@ugto.mx; ffernand@inf.uc3m.es;
†These authors contributed equally to this work.

Abstract

Reward Machines provide a structured representation of sequential tasks by decomposing them into intermediate subtasks. Although multiple RM decompositions can encode the same task, they often yield substantially different learning behavior. Understanding why these differences arise remains challenging because evaluation measures only overall performance and does not reveal where learning difficulty is concentrated. In this paper, we introduce a methodology that combines independent RM-state training with end-to-end evaluation to characterize both local subtask learning and overall task performance. Independent training identifies learning bottlenecks within individual RM states, whereas end-to-end evaluation shows how these bottlenecks affect overall learning behavior. Experiments in MiniGrid DoorKey and VizDoom Deadly Corridor show that different Reward Machine decompositions yield distinct learning dynamics, even when they encode the same task objective. Our results demonstrate that learning bottlenecks arise at specific RM states and explain the performance differences observed across decompositions. These findings provide practical insight into the

analysis and design of Reward Machines and highlight the importance of treating task decomposition as a design variable rather than a fixed task specification.

Keywords: Reinforcement Learning, Reward Machines, Task Decomposition, Learning Dynamics, Long-Horizon Tasks

1 Introduction

In Reinforcement Learning (RL), tasks with long-term horizons are challenging because rewards often arrive long after the relevant decisions are made, making it difficult to determine which actions are responsible for the outcome. Previous work has addressed this challenge through temporal abstraction and hierarchical problem decomposition [1–4]. However, tasks that share the same objective may still require different learning strategies depending on their underlying structure. While some tasks can be solved with only a few decisions, others require long sequences of coordinated actions to achieve the desired outcome. This paper studies this idea using Reward Machines (RMs). RMs are formal models that encode task progression through events, allowing rewards to depend on prior outcomes [5, 6]. RMs can improve learning by providing intermediate rewards, helping the agent learn from situations beyond direct experience, and expressing tasks in a structured, understandable way [7–9]. We focus on the level of detail used to represent tasks in RMs, namely, how a complex objective is decomposed into intermediate stages. Simpler RMs provide less guidance during learning, whereas more detailed RMs can introduce intermediate objectives that speed up learning but also increase the number of sequential dependencies. Therefore, task details do not necessarily make learning easier or harder; rather, they shift where difficulty arises. For example, in a robotic pick-and-place task, learning and exploration may be more difficult when rewards are assigned only after the final placement is achieved. Under a more detailed RM, rewards may be assigned to intermediate events such as reaching the object, grasping it, and transporting it. Although learning can be improved by including intermediate objectives, longer action sequences are also required, during which errors may prevent task completion. Therefore, finer task decompositions provide more feedback but also increase susceptibility to sequential failures. Although RM have primarily been studied as tools for improving learning efficiency [5, 6], the learning of RM structures from data, hierarchical organizations of RMs [10, 11], and multiagent RM settings [12, 13] have also been explored in previous work. However, the RM structure is often treated primarily as a design choice. As a result, an important aspect of long-horizon tasks is overlooked: two RMs that describe the same objective may still induce very different learning conditions because of differences in feedback frequency, sequential dependencies, and opportunities for failure. We hypothesize that the way a task is decomposed into Reward Machine subtasks influences where learning difficulties arise. Different task decompositions redistribute the learning effort across sequential subtasks, each with its own level of learning complexity. Consequently, some subtasks may require substantially more training than others, creating learning bottlenecks that delay or prevent the successful completion

of subsequent stages. Identifying where these bottlenecks occur is relevant not only for understanding learning behavior but also for determining whether a task decomposition is likely to increase training cost or reduce learning efficiency. This motivates the central question of this work: how does increasing task detail affect where learning bottlenecks emerge, and can these patterns be used to evaluate or improve Reward Machine decompositions before full training is required? To answer this question, we analyze RM structure as a design variable and examine how different task decompositions affect learning speed, stability, and successful task completion, and when increasing task detail improves learning or instead reduces performance in sequential decision-making.

The main contributions of this paper are as follows:

- An analysis of task decomposition in Reward Machines that examines how increasing task detail shifts where learning becomes more difficult rather than simply making learning easier.
- An evaluation of how varying levels of RM detail affect agent performance, using success rate and the number of steps required to complete subtasks as measures of difficulty.
- Identification of subtasks that become harder to learn when additional intermediate stages are introduced, requiring more episodes to achieve consistent behavior.
- Examination of how difficulties in early subtasks affect later stages, leading to accumulated errors during task execution.
- Empirical evidence shows that tasks with the same final objective can produce different learning behaviors depending on how they are decomposed into subtasks.

2 Related Work

Several research directions have addressed long-horizon RL by decomposing tasks, using temporal abstractions, and employing structured task representations. Table 1 summarizes the research directions most closely related to our work and highlights the key distinctions between them and our approach.

Hierarchical Reinforcement Learning (HRL) and Learned Hierarchies are two research directions within HRL that improve exploration and learning efficiency through temporal abstractions and hierarchical policies [1, 2, 14]. Building on this idea, learned hierarchy methods automatically discover reusable skills and behavioral abstractions rather than relying on manually designed decompositions [15–19]. Both directions emphasize constructing effective task decompositions that facilitate learning. In contrast, this work examines how different task decompositions affect the learning process and identifies where learning difficulties arise. RMs and Their Extensions constitute a research direction in which RMs are represented as finite-state automata that encode reward-relevant event sequences, thereby providing structured reward representations that can improve reward shaping, credit assignment, and policy learning [5, 6]. This representation has been successfully applied to increasingly complex domains, including hierarchical tasks, noisy observations, robotic manipulation, and multiagent environments [10–13, 20–22]. Despite this broad applicability, existing work typically represents each task with a single RM decomposition. In contrast, this

Table 1: Comparison of related approaches and their relationship to our work.

Area	Representation used	Primary purpose	Limitation relative to our work
Hierarchical RL [1, 2, 14]	Hierarchical task decomposition	Improve exploration and learning efficiency.	Assumes a predefined task decomposition rather than comparing alternatives.
Learned hierarchies [15–19]	Automatically learned abstractions	Learn reusable skills or subpolicies	Focuses on discovering task decompositions rather than comparing them.
Reward Machines [5–8]	Automata-based task decomposition	Reward shaping and policy learning	Each task is decomposed using a single RM structure.
Extended RM settings [10–13, 20–23]	Hierarchical, noisy-label, and multiagent RMs	Improve applicability to complex environments.	Focuses on extending RM applications rather than analyzing various decompositions.
Curriculum learning [24–27]	Ordered task progression	Accelerate learning	Changes the order of tasks rather than their decomposition.
Temporal logic and automata RL [28, 29]	Formal sequential task specifications	Represent long-horizon objectives	Specifies tasks but does not compare alternative decompositions.
This work	Reward Machine decomposition	Understand where learning difficulties arise.	Requires training to evaluate the quality of a task decomposition.

work investigates the influence of alternative task decompositions on learning behavior. Curriculum Learning and Task Specification are two research directions in which curriculum learning organizes experiences in order of increasing difficulty [24–27]. At the same time, long-horizon objectives are formally specified through logical task descriptions using temporal logic and automata-based approaches [28, 29]. Although both research directions provide structured frameworks for facilitating learning or specifying sequential objectives, the effects of different task decompositions on the learning process remain unexamined. Instead, this work uses Reward Machine decompositions to identify where learning difficulties emerge within sequential tasks.

3 Background

RL [30] studies how an agent learns to make decisions through interactions with an environment, typically modeled as a Markov Decision Process (MDP). An MDP is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ denotes the transition dynamics, $R(s,a)$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. At each time step t, the agent observes a state s_t, selects an action a_t according to a policy $\pi(a|s)$, receives a reward r_t, and transitions to a new state s_{t+1}. The objective is to learn a policy that maximizes the expected discounted return $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

RMs [5] provide a formal framework for representing non-Markovian reward functions with finite-state automata. An RM is defined as $\mathcal{R} = \langle U, u_0, F, \delta_u, \delta_r \rangle$, where U is a finite set of RM states, $u_0 \in U$ is the initial state, $F \subseteq U$ is a set of terminal (accepting) states, and $\mathcal{P}$ is a finite set of atomic propositions (or symbolic events) that describe relevant high-level occurrences in the environment. The transition function $\delta_u : U \times 2^{\mathcal{P}} \to U$ updates the RM state based on the observed propositions, while $\delta_r : U \times 2^{\mathcal{P}} \to \mathbb{R}$ defines the corresponding rewards. A trajectory is successful if the RM reaches a terminal state $u \in F$. Thus, the RM defines task completion in terms of

the sequence of observed events rather than individual transitions. A labeling function $L : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to 2^{\mathcal{P}}$ maps environment transitions to sets of atomic propositions that hold after each transition. A label trace is a sequence $\lambda = (l_0, l_1, \dots)$, where each label $l = L(s, a, s')$ corresponds to the observed high-level event. RM can be interpreted as functions over such traces, allowing rewards to depend on temporally extended properties of the agent's behavior. Given a label trace λ, the RM processes it incrementally by updating its state as $u' = \delta_u(u, l), \quad r = \delta_r(u, l)$. The interaction between the MDP and the RM induces a product MDP with state space $\mathcal{S} \times U$, in which the reward becomes Markovian: $R'((s, u), a, (s', u')) = \delta_r(u, L(s, a, s'))$. This construction effectively transforms a non-Markovian reward function defined over traces into a Markovian reward function over the augmented state space $\mathcal{S} \times U$, enabling the use of standard RL algorithms.

Q-Learning for Reward Machines (QRM) is a natural way to leverage the RM structure by learning an action-value function for each RM state: $\{Q_u(s, a)\}_{u \in U}$. Given a transition (s, a, s') with label $l = L(s, a, s')$, the RM transitions to $u' = \delta_u(u, l)$ and produces reward $r = \delta_r(u, l)$. The Q-learning update is: $Q_u(s, a) \leftarrow (1 - \alpha)Q_u(s, a) + \alpha\,(r + \gamma \max_{a'} Q_{u'}(s', a'))$. This formulation is equivalent to learning a value function on the product MDP with state (s, u).

Deep Q-Learning for Reward Machines (DQRM) replaces QRM's tabular action-value functions with neural network approximations, enabling learning in high-dimensional state spaces. In our implementation, the resulting action-value functions are approximated using Double Deep Q-Networks (DDQN).

4 Problem Formulation

Building on the RL and Reward Machine framework introduced in section 3, we study how the design of task decompositions affects learning performance in complex environments. Task Decomposition with RM provides a structured way to represent a task as a sequence of subtasks. Each RM state $u \in U$ can be interpreted as a subtask with a specific objective and a corresponding action-value function Q_u. In this work, we consider the problem of designing RM-based decompositions for a given task. In particular, instead of assuming a fixed RM, we allow multiple candidate decompositions, each defined by a different RM structure: $\mathcal{R}^{(i)} = \langle U^{(i)}, u_0^{(i)}, F^{(i)}, \delta_u^{(i)}, \delta_r^{(i)} \rangle$.

4.1 Metrics

Each decomposition induces a distinct product MDP and therefore a distinct learning problem. Each RM state is evaluated using two complementary measures: the probability that its associated subtask is completed and the efficiency with which the subtask is executed once successful completion is achieved. Formally, for each RM state $u \in U$, the following quantity is defined: $\mathbf{m}(u) = \big(\mathrm{Succ}(u), \mathrm{Qual}(u)\big)$, where $\mathrm{Succ}(u)$ denotes the probability that the subtask associated with u is completed, and $\mathrm{Qual}(u)$ denotes the execution efficiency conditioned on successful completion, measured by the number of environment steps required to complete the subtask. Although domain- and environment-specific measures may also characterize execution quality, our analysis focuses exclusively on execution efficiency to enable consistent comparisons across all

task decompositions and environments considered in this study. At each episode i, we define a binary success indicator:

$$s_i(u) = \begin{cases} 1 & \text{if } u \text{ is successfully completed in episode } i, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Additionally, if the subtask is completed, we record a quality measure $q_i(u)$ for the execution (e.g., episode duration). This measure is defined only for successful episodes. Given a window of N episodes, empirical estimates of completion frequency and execution efficiency for each subtask are computed:

$$\text{Succ}(u) = \frac{1}{N} \sum_{i=1}^{N} s_i(u) \tag{2}$$

$$\text{Qual}(u) = \frac{1}{\sum_{i=1}^{N} s_i(u)} \sum_{i=1}^{N} s_i(u) q_i(u) \tag{3}$$

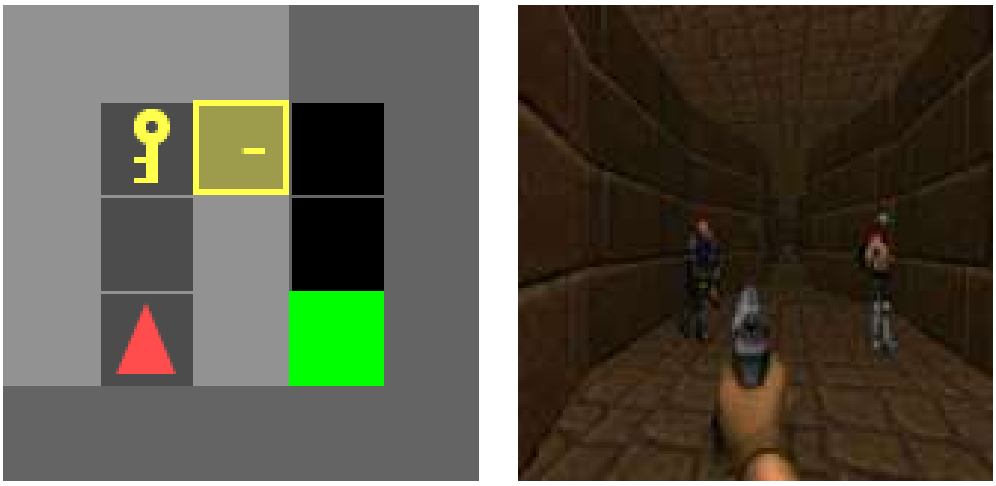
where $s_i(u) \in \{0, 1\}$ indicates whether subtask u was successfully completed in episode i, and $q_i(u)$ is the corresponding execution efficiency. If no successful executions are observed within the window, $\text{Qual}(u)$ is set to 0.

4.2 Task decomposition with reward machine

A key design trade-off in task decomposition is choosing the appropriate level of granularity. A task can be represented by RMs at different levels of granularity, ranging from compact representations with few RM states to more detailed representations with many intermediate states. As granularity increases, individual subtasks may be easier to learn because additional intermediate objectives provide guidance. However, the number of sequential learning stages also increases, making successful task completion more dependent on the successful execution of a longer sequence of subtasks. Consequently, increasing the level of detail does not necessarily make learning uniformly easier or harder. Instead, the distribution of learning difficulty across the task changes. As a result, different RM decompositions induce distinct subtask performance profiles that reflect where learning bottlenecks are likely to emerge. Given an environment modeled as an MDP $\mathcal{M}$ and a set of candidate RM $\mathcal{R}^{(i)}$, the objective of this work is to understand how different task decompositions influence learning. To this end, a policy $\pi^{(i)}$ is learned for each decomposition, and the corresponding subtask performance profile $\mathbf{m}^{(i)}(u)$ is estimated. These profiles are then used to analyze how overall learning performance is affected by the structure and granularity of each decomposition. By treating task decomposition as a design variable rather than a fixed component, this formulation enables a systematic study of the relationship between Reward Machine decompositions and learning behavior.

4.3 Domain Examples

We evaluate our approach in two benchmark environments that represent different learning challenges: a combat task in VizDoom and a navigation task in MiniGrid (Figure 1). In the VizDoom Deadly Corridor task, the agent must defeat a sequence of enemies before reaching the final objective, the armor. This requires combining perception, navigation, aiming, shooting, and survival while progressing through the environment. Performance is measured by the success rate, defined as the probability of completing the task. In the MiniGrid DoorKey task, the agent must reach the goal by first collecting a key and opening a door. Performance is measured by both the success rate and the number of execution steps required to complete the task. These environments provide complementary evaluation scenarios: VizDoom emphasizes sequential decision-making in a dynamic combat environment, whereas MiniGrid focuses on efficient navigation in a structured environment. These environments allow us to evaluate how different task decompositions perform across distinct learning challenges, as discussed in Section 6.



(a) MiniGrid DoorKey. (b) VizDoom Deadly Corridor.

Fig. 1: Experimental domains used in this work.

5 Method

This section presents the methodology for analyzing how Reward Machine (RM) decompositions influence learning behavior. Rather than evaluating task performance only at the global level, the proposed approach examines the learning dynamics associated with individual RM states. To this end, two complementary training protocols are introduced: an independent subtask training protocol that characterizes subtask difficulty and identifies bottlenecks, and an end-to-end training protocol that evaluates

the overall effect of different RM decompositions. By combining these protocols, the effects of decomposition granularity, subtask difficulty, and error accumulation across intermediate states in long-horizon tasks can be studied. An overview of the proposed methodology is shown in Figure 2, which illustrates the relationships among the training protocols, performance measures, and analysis stages used throughout this work. Starting from the original task MDP, candidate Reward Machine decompositions are generated and learned. Independent subtask training and end-to-end training evaluate state-level learning behavior and the effect of decomposition granularity on task performance, respectively. Subtask success, subtask quality, and cumulative reward enable diagnostic analyses of error accumulation, bottleneck identification, and granularity, providing methodological insights for iteratively refining the candidate Reward Machine decomposition.

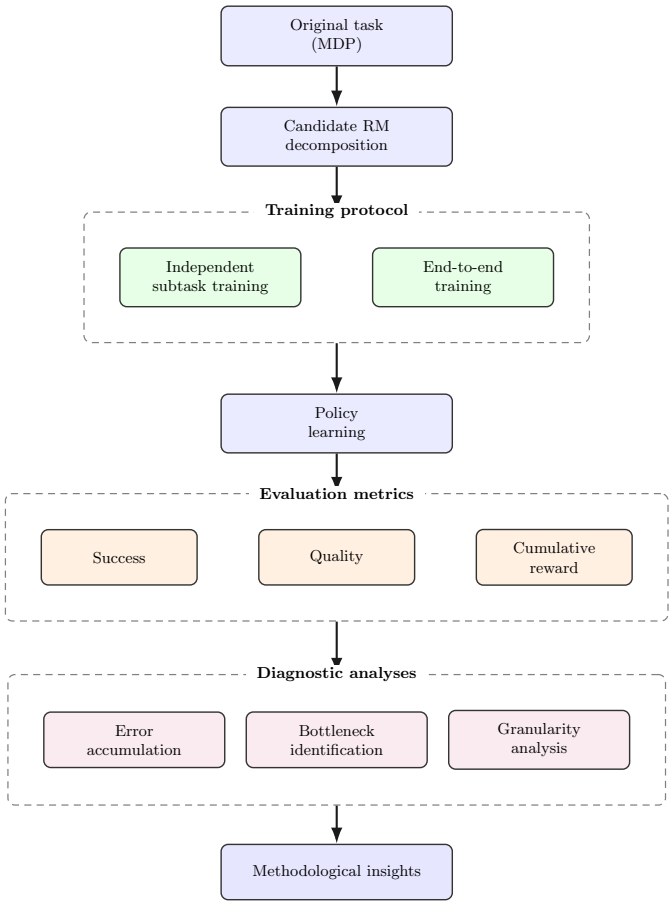


Fig. 2: Conceptual overview of the proposed methodology.

5.1 Subtask-Centric Learning

In end-to-end training, the collection of action-value functions associated with RM states is optimized as the complete task is solved. Consequently, performance is typically evaluated using global metrics such as cumulative reward. However, these aggregate measures provide only limited insight into the internal learning dynamics of the decomposition. In particular, they do not identify which subtasks are more difficult to learn, reveal how local failures affect subsequent stages, or determine whether a given RM structure introduces bottlenecks that limit overall performance. In this work, RMs are used not only as a task-decomposition mechanism but also as an analytical framework for studying subtask difficulty and error accumulation. To this end, two complementary training protocols are considered: independent subtask training and end-to-end training. The combination of these protocols provides both local and global perspectives on the learning dynamics induced by the decomposition.

Given an RM decomposition, a separate action-value function can be associated with each non-terminal RM state. Thus, the set of value functions is defined as $\mathcal{Q} = \{Q_{u_0}, Q_{u_1}, \ldots, Q_{u_n}\}$. Each action-value function Q_u specializes in learning the behavior needed to progress from its corresponding RM state u to the next relevant RM state, according to the RM transition structure. For example, in the DoorKey decomposition, the task is represented by three subtasks: obtaining the key, opening the door, and reaching the goal. Consequently, the decomposition yields the following specialized value functions: $Q_{u_0} \rightarrow$ key acquisition, $Q_{u_1} \rightarrow$ door opening, $Q_{u_2} \rightarrow$ goal reaching. Each episode begins with the original environment distribution, including randomized agent, key, and door positions. Therefore, subtasks are learned under the same variability encountered during normal task execution rather than from manually initialized, subtask-specific states. We introduce an independent subtask training protocol to study how increasing task detail affects learning for individual subtasks. Rather than learning all RM states simultaneously, the protocol trains each RM state independently, allowing the success probability of each subtask to be analyzed in isolation. Let $U = \{u_0, u_1, \ldots, u_n\}$ denote the ordered sequence of non-terminal RM states. Training proceeds sequentially. First, the value function for u_0 is trained until it consistently triggers the first RM transition. The resulting value function is then frozen and used to reach the next RM state, where the corresponding value function is trained. This procedure is repeated for every non-terminal RM state. Algorithm 1 formally specifies the independent subtask training procedure. Figure 3 complements the algorithm by illustrating the overall workflow, including subtask selection, independent training, policy evaluation, and the freezing procedure before progressing to the next RM state.

To ensure a fair comparison across subtasks, each RM state, representing a different subtask, is trained with the same number of episodes. An episode is one complete attempt to solve the associated subtask. Consequently, any observed performance differences can be attributed to subtask difficulty rather than unequal training exposure. Because every subtask receives the same training budget, the learning dynamics of individual subtasks can be analyzed independently. This enables a direct comparison of subtask difficulty, helps identify bottlenecks in the task decomposition, and reveals how learning difficulty is distributed across RM states. The second protocol uses an end-to-end training setting. In contrast to the independent subtask protocol, learning

Algorithm 1 Independent Subtask Training

Require: Reward Machine $\mathcal{R} = (U, u_0, F, \delta_u, \delta_r)$
Require: Labelling function $L(s, a, s')$
Require: Number of runs M
Require: Number of episodes N
Require: Set of previously selected frozen value functions $\mathcal{Q}^*_{\text{frozen}}$
Require: SubtaskCompleted(u, ℓ) returns true if event ℓ triggers a transition from RM state u to a successor RM state

$\mathcal{Q}^* \leftarrow \mathcal{Q}^*_{\text{frozen}}$
for all non-terminal RM states $u_i \in U \setminus F$ **do**
  for $m \leftarrow 1$ **to** M **do**
    Initialize a Double DQN agent $Q^{(m)}_{u_i}$ with random weights
    for *episode* $\leftarrow 1$ **to** N **do**
      Reset the environment and obtain the initial state $s \leftarrow s_0$
      Set $u \leftarrow u_0$
      if $u \neq u_i$ **then**
        Execute the frozen value functions in $\mathcal{Q}^*$ sequentially until the current RM state becomes u_i or the episode terminates
      end if
      if the current RM state is u_i **then**
        completed $\leftarrow$ false
        while episode not terminated and not completed **do**
          Select an action a from the current state s using the policy induced by $Q^{(m)}_{u_i}$
          Execute action a and observe the next environment state s'
          Compute the event $\ell \leftarrow L(s, a, s')$
          Compute the reward $r \leftarrow \delta_r(u, \ell)$
          Compute the next RM state $u' \leftarrow \delta_u(u, \ell)$
          Update $Q^{(m)}_{u_i}$ with transition (s, a, r, s')
          completed $\leftarrow$ SubtaskCompleted(u, ℓ)
          Update the current environment state $s \leftarrow s'$
          Update the current RM state $u \leftarrow u'$
        end while
      end if
    end for
    Store the trained agent $Q^{(m)}_{u_i}$ for later evaluation
  end for
  Evaluate the trained agents $\{Q^{(1)}_{u_i}, \ldots, Q^{(M)}_{u_i}\}$ using their success rates and execution steps
  Select $Q^*_{u_i}$ as the agent with the highest success rate; in case of a tie, select the agent with the fewest execution steps
  Freeze the selected agent $Q^*_{u_i}$
  Add $Q^*_{u_i}$ to $\mathcal{Q}^*$
end for
return $\mathcal{Q}^*$

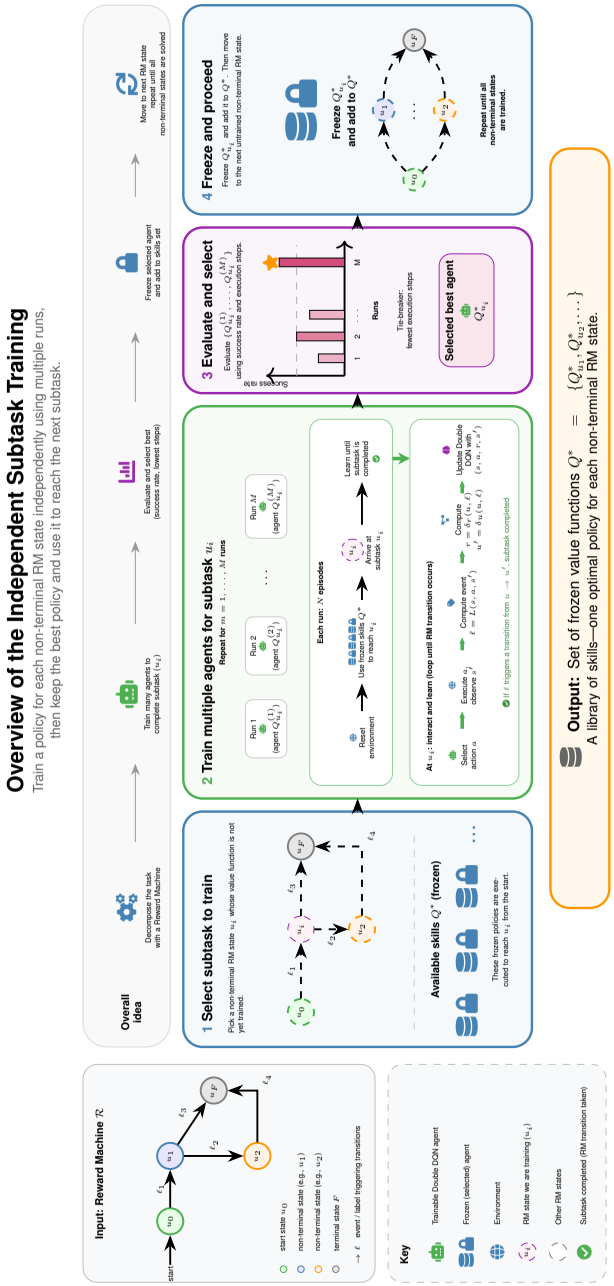


Fig. 3: Overview of the proposed Independent Subtask Training protocol.

occurs across the entire task, starting from the initial environment distribution as the agent progresses through the Reward Machine states. At each interaction step, the current RM state determines the active value function Q_u used for action selection. As symbolic events are observed, the RM transitions according to its transition function, activating the value function associated with the new RM state. Consequently, learning is distributed across the sequence of subtasks induced by the RM while maintaining a single continuous interaction with the environment. This second protocol evaluates the overall effect of different task decompositions by comparing the goal-only, coarse-grained, and fine-grained RM, with and without RM, under the same total training budget. The two training protocols provide complementary perspectives on RM-based task decompositions. Independent subtask training identifies where learning difficulties arise within the decomposition, whereas end-to-end training evaluates how those difficulties affect overall task performance. Algorithm 2 formally specifies the end-to-end training procedure. Figure 4 complements the algorithm by providing a conceptual overview of the training workflow and illustrating how the active value function changes as the Reward Machine transitions between states.

Algorithm 2 End-to-End Training

Require: Reward Machine $\mathcal{R} = (U, u_0, F, \delta_u, \delta_r)$
Require: Labelling function $L(s, a, s')$
Require: Number of episodes N
Initialize Double DQN agents $\mathcal{Q} = \{Q_u \mid u \in U \setminus F\}$ with random weights
for *episode* $\leftarrow 1$ **to** N **do**
    Reset the environment and obtain the initial state $s \leftarrow s_0$
    Set the current RM state $u \leftarrow u_0$
    while episode not terminated **do**
        Select an action a from the current environment state s using the policy induced by the agent Q_u
        Execute action a and observe the next environment state s'
        Compute the event $\ell \leftarrow L(s, a, s')$
        Compute the reward $r \leftarrow \delta_r(u, \ell)$
        Compute the next RM state $u' \leftarrow \delta_u(u, \ell)$
        Update the agent Q_u using the Double DQN update rule with transition (s, a, r, s')
        Update the current environment state $s \leftarrow s'$
        Update the current RM state $u \leftarrow u'$
    end while
end for
return $\mathcal{Q}$

A bottleneck is a subtask whose learning difficulty limits the performance of subsequent RM states and, consequently, the performance of the complete task. To characterize the learning difficulty of each subtask, the success rate and execution steps introduced in Section 4 are analyzed jointly. A success rate alone is insufficient,

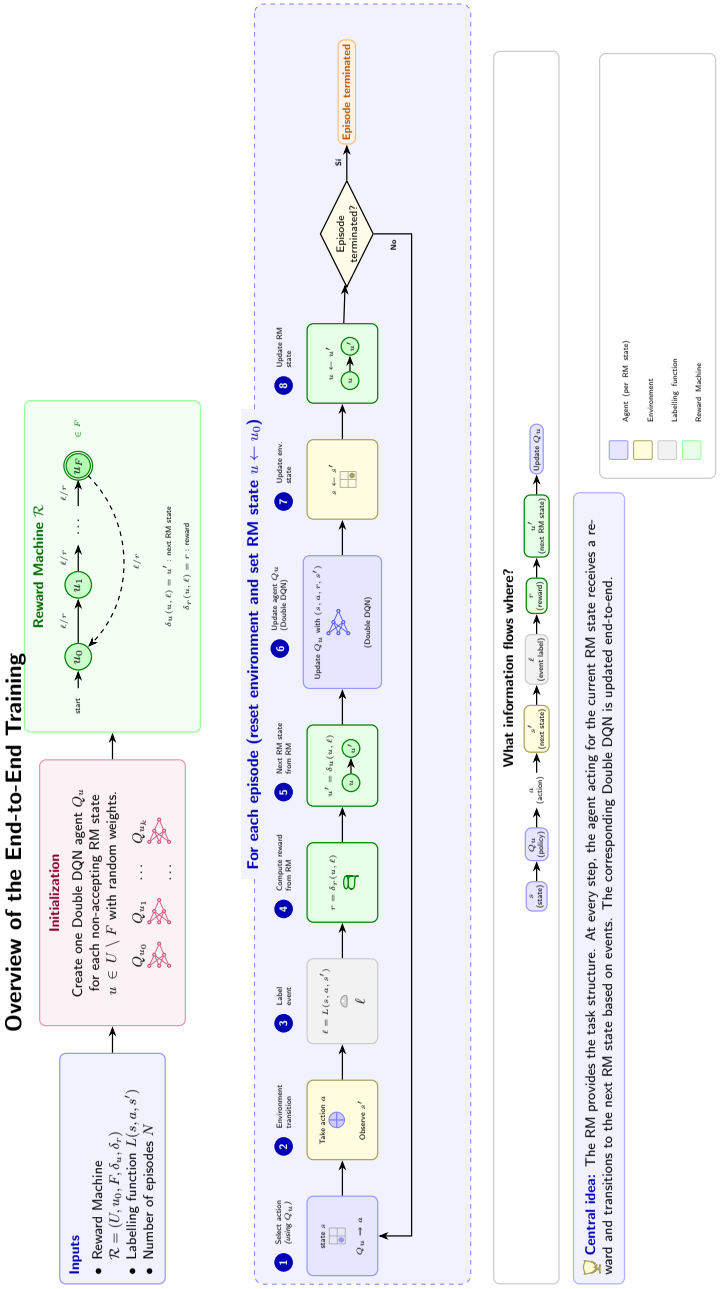


Fig. 4: Overview of the proposed End-to-End Training protocol.

since a subtask may be completed reliably yet still require substantially more execution steps than the remaining subtasks. Therefore, both metrics are evaluated under identical training conditions to provide a more complete characterization of the learning behavior associated with each RM state. Potential bottlenecks are identified by comparing the relative learning difficulty across subtasks. Subtasks that consistently require greater learning effort than the remaining RM states are considered potential bottlenecks. Their localization at the RM-state level is achieved through the independent subtask training protocol. In contrast, their effect on overall task performance is assessed through the end-to-end training protocol. By combining these complementary analyses, local learning limitations can be distinguished from their global impact on the complete task.

6 Experimental Results

We evaluate the proposed Reward Machine (RM) analysis methodology on two domains: MiniGrid DoorKey and VizDoom Deadly Corridor. The goal of the experimental evaluation is not only to compare final task performance but also to analyze how the structure of a Reward Machine redistributes learning difficulty across subtasks. The experiments are organized around three research questions:

- **RQ1:** Where does learning difficulty emerge inside a Reward Machine decomposition?
- **RQ2:** How does the design of a Reward Machine influence end-to-end learning?
- **RQ3:** Do these learning patterns remain consistent across different domains?

These research questions investigate learning at two complementary levels. Independent subtask training identifies where learning difficulties arise within the Reward Machine, while end-to-end training assesses how these local difficulties affect overall performance on the complete task. Thus, end-to-end evaluation indicates which RM design performs better, whereas independent RM-state training helps explain why those differences arise.

6.1 Experimental Protocol

Each RM state is implemented as a fully connected DDQN with two hidden layers of sizes 512 and 256. Training uses mini-batches of 128 samples, a learning rate of 5×10^{-5}, a discount factor of $\gamma = 0.6$, and a soft target update parameter of $\tau = 0.001$. Exploration follows a softmax policy with a temperature that decays linearly throughout training, gradually shifting from exploration to exploitation. For the independent subtask training protocol, each RM state is trained for 300k episodes across 5 independent runs. The end-to-end training protocol is trained for 100k episodes using the same number of runs. For the VizDoom experiments, each RM state is trained for 6k episodes across 10 independent runs. Reported results correspond to the mean and standard deviation across runs ($\mu \pm \sigma$).

6.2 RQ1: Where does learning difficulty emerge inside a Reward Machine decomposition?

We first investigate where the learning difficulty arises in the DoorKey task. To this end, we execute algorithm 1. By training each Reward Machine (RM) state independently, we isolate the learning process for each subtask and pinpoint where the learning effort is concentrated. In contrast, end-to-end training only reveals overall task performance, making it difficult to determine which stage is responsible for the observed learning difficulty.

6.2.1 Success Rate

Figure 5 shows that the three DoorKey subtasks are not equally difficult to learn. Key acquisition reaches a high success rate early in training, whereas door opening requires substantially more experience to achieve comparable performance. Once these prerequisite behaviors are acquired, the agent also learns to reliably reach the goal. This trend holds across all five independent runs. Key acquisition and goal reaching converge faster and with lower variability, whereas door opening converges more slowly and shows greater variability throughout training. These results identify door opening as the primary learning bottleneck in the DoorKey Reward Machine decomposition.

6.2.2 Execution Quality

Success rate indicates whether a subtask is completed, whereas execution quality measures how efficiently it is completed, as reflected in the number of environment steps required. Lower step counts, therefore, indicate more efficient policies. Figure 6 shows the evolution of execution quality for each RM state during training. Consistent with the success-rate analysis, the door-opening state requires substantially more interaction with the environment before efficient policies emerge. This indicates that the learning bottleneck is reflected not only in the ability to solve the subtask but also in the amount of experience required to solve it efficiently.

The two metrics provide complementary evidence. Both consistently identify the door-opening state as the primary learning bottleneck, indicating that this conclusion is robust across evaluation metrics. Overall, RQ1 is answered positively: independent RM-state training successfully identifies where learning difficulty arises within the DoorKey Reward Machine decomposition. Having identified the door-opening state as the primary bottleneck, we next investigate whether alternative Reward Machine decompositions can mitigate its impact on end-to-end learning.

6.3 RQ2: How does the design of a Reward Machine influence end-to-end learning?

Having identified where learning difficulties arise during independent subtask training, we now investigate whether task decomposition influences end-to-end learning. To answer this question, we compare several Reward Machine (RM) representations of the DoorKey task, ranging from a goal-only RM to a fine-grained decomposition, along with two intermediate coarse-grained variants. A baseline without RM structure,

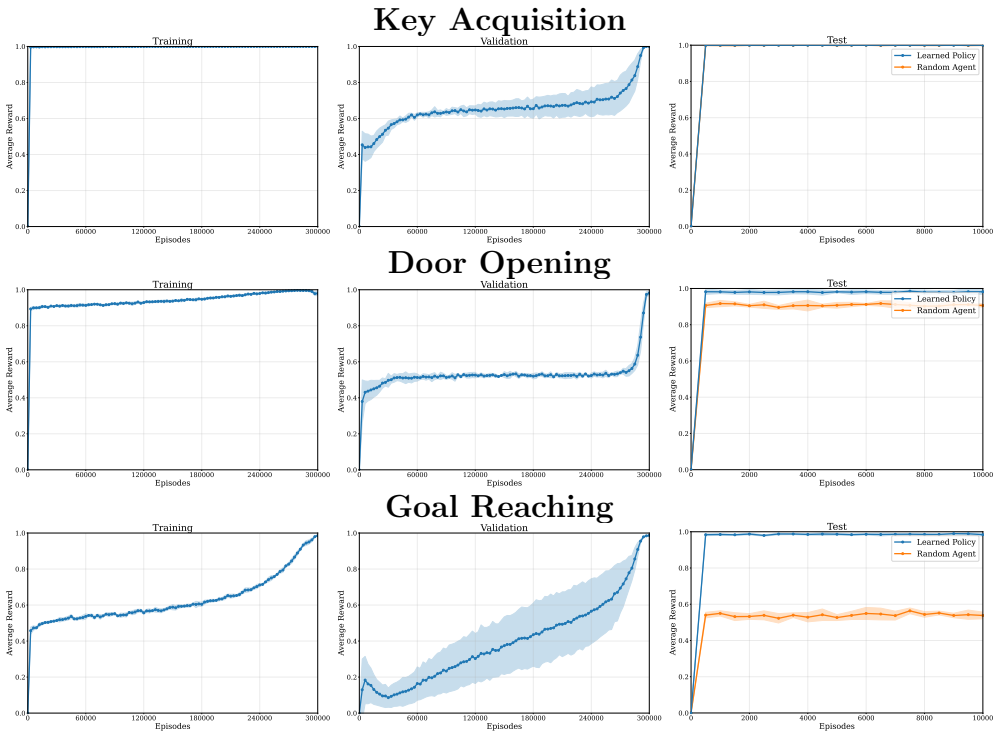


Fig. 5: Learning dynamics of the fine-grained Reward Machine during independent subtask training in the DoorKey environment with softmax exploration. Rows correspond to Reward Machine states, and columns report training, validation, and test performance.

with intermediate rewards and a random policy, is included for comparison. Figure 7 illustrates the corresponding Reward Machine decompositions. Although all representations describe the same underlying task, they differ in the level of decomposition. The goal-only RM offers the simplest representation, the fine-grained RM decomposes the task into three sequential subtasks, and the coarse-grained RMs provide intermediate levels of abstraction by combining parts of this structure.

Figure 8 reports the average number of environment steps required to complete the full task.

All Reward Machine representations encode the same DoorKey objective yet yield different end-to-end learning behaviors. Among them, the coarse-grained representations require the fewest environment steps to complete the task, whereas both the fine-grained and goal-only representations require more interaction. These results demonstrate that the way a task is decomposed directly affects end-to-end learning performance. The independent subtask analysis from RQ1 explains these differences. There, the door-opening stage was identified as the primary learning bottleneck because it required substantially more training than the remaining subtasks. The fine-grained RM preserves this bottleneck as a separate intermediate state, whereas

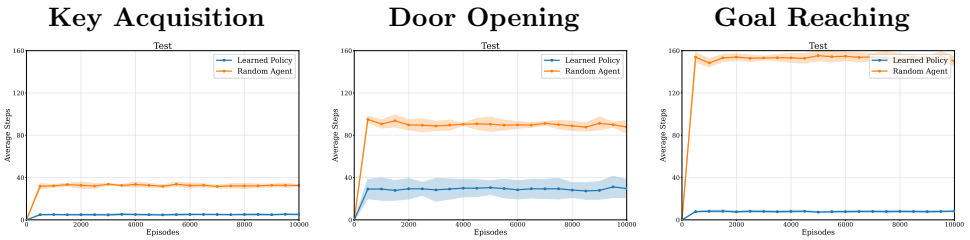
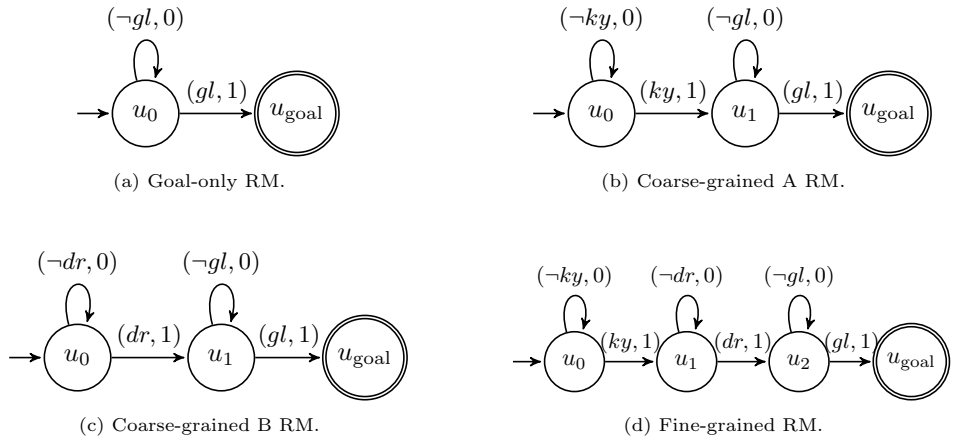


Fig. 6: Execution efficiency of independent subtask training in the DoorKey environment with softmax exploration. Curves show the average number of environment steps required to complete each subtask.



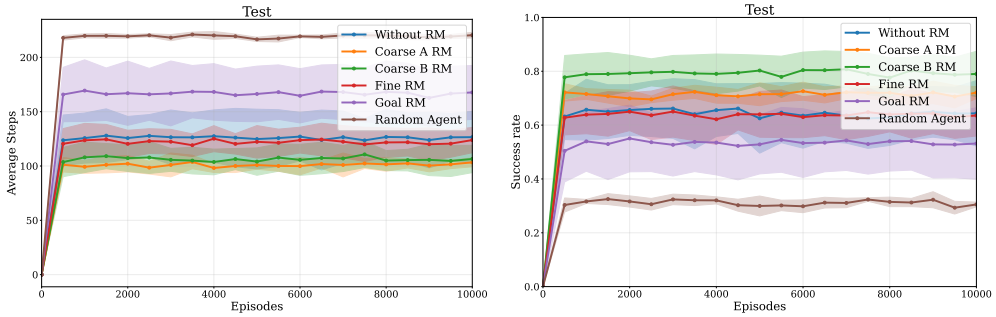
(a) Goal-only RM.

(b) Coarse-grained A RM.

(c) Coarse-grained B RM.

(d) Fine-grained RM.

Fig. 7: Alternative Reward Machine representations of the MiniGrid DoorKey task at different levels of task decomposition. Each Reward Machine encodes the same overall objective and is used for end-to-end evaluation.



(a)

(b)

Fig. 8: End-to-end learning dynamics for the Reward Machine representations shown in Fig. 7. (a) Average number of environment steps required to complete the task ($\mu \pm \sigma$). (b) Probability of successfully completing the task ($\mu \pm \sigma$).

the coarse-grained representations absorb part of this structure into larger subtasks, redistributing the learning effort. Consequently, the bottleneck identified during independent training is reflected in the end-to-end learning behavior. Interestingly, the two coarse-grained decompositions do not produce identical behavior, despite having the same level of abstraction. While both reduce the number of intermediate RM states, they merge different subtasks and consequently achieve different performance. This indicates that the effectiveness of a decomposition depends not only on the number of subtasks but also on how the original task is partitioned. In particular, the decomposition adopted by Coarse A consistently yields better performance than that of Coarse B, suggesting that the specific grouping of subtasks within the Reward Machine influences the resulting learning dynamics. The goal-only RM represents the opposite extreme of the task decomposition spectrum. By removing almost all intermediate structure, it provides little guidance during learning and performs worse than the remaining Reward Machine representations. These results suggest that effective Reward Machine design requires an appropriate balance in task decomposition. Too little structure provides insufficient guidance, whereas excessive decomposition may isolate difficult subtasks, thereby amplifying learning bottlenecks. These findings show that effective Reward Machine design is not simply a matter of choosing the number of intermediate states. Rather, the key consideration is how a particular decomposition redistributes learning effort across the sequential tasks. Independent RM-state training provides this insight by revealing which subtasks dominate the learning process and are therefore most likely to become learning bottlenecks. Overall, RQ2 is answered in the affirmative: different Reward Machine decompositions of the same task produce distinct end-to-end learning behaviors. The independent subtask analysis from RQ1 explains why these differences arise by identifying how each decomposition redistributes learning effort across the task. In this sense, end-to-end training reveals which Reward Machine design performs better, whereas independent RM-state training explains why.

6.4 RQ3: Do these learning patterns remain consistent across different domains?

Having shown that learning bottlenecks can be identified in DoorKey, we now investigate whether the same phenomenon also arises in a more complex environment. To answer this question, we evaluate our methodology on VizDoom Deadly Corridor, a domain that combines navigation and combat and is substantially more challenging than DoorKey. Figure 9 illustrates the VizDoom environment and the Reward Machine used in this experiment. The agent must eliminate enemies while progressing through the corridor. Rather than modeling the full combat sequence, we represent the task with a simplified Reward Machine composed of two sequential combat subtasks. Although the original scenario contains six enemies, this abstraction preserves the sequential structure needed to analyze how learning difficulty evolves throughout the task. This simplification aligns with the experiment's objective. Our goal is not to reproduce the full combat sequence in detail, but to identify where learning difficulty first emerges. If independent RM-state training can reveal bottlenecks in this visually

complex combat domain, it would indicate that the proposed methodology generalizes beyond the navigation-based DoorKey environment.

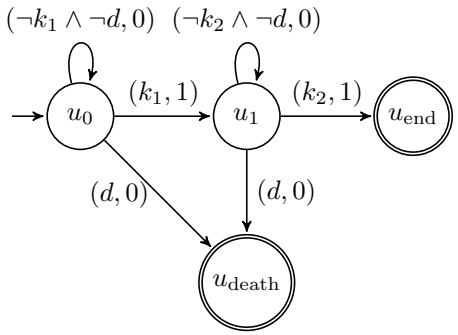


Fig. 9: Reward Machine representation of the sequential elimination task.

Using this representation, we independently train each RM to identify where learning difficulty arises throughout the task. To determine whether a combat subtask has been learned, we evaluate its success rate during independent training. Unlike DoorKey, the number of environment steps is not an appropriate measure of execution quality in this domain. An agent may eliminate an enemy quickly while taking excessive damage or wasting ammunition, so fewer steps do not necessarily indicate better performance. Consequently, success rate is the most reliable indicator of whether each combat subtask has been learned. Figure 10 shows the success rate achieved by each RM state during independent subtask training.

The results show that the proposed methodology can also identify localized learning bottlenecks in the more complex VizDoom environment. Unlike DoorKey, where the primary bottleneck occurs during the intermediate door-opening stage, Deadly Corridor exhibits its main learning difficulty during the initial combat stage. This observation aligns with the characteristics of the first RM state, in which the agent must simultaneously perceive the environment, navigate, aim, shoot, and survive before progressing to the next subtask. Although the bottleneck appears at a different point in the task, the underlying learning behavior remains consistent across both domains. Independent RM-state training successfully identifies where learning difficulty is concentrated, indicating that the proposed methodology captures properties of the task decomposition rather than characteristics specific to a particular environment. Overall, RQ3 is answered in the affirmative: independent RM-state training identifies where learning difficulty emerges in both DoorKey and VizDoom, despite bottlenecks occurring at different stages of each task. These results demonstrate that the proposed analysis generalizes beyond a single environment and provides a consistent framework for studying how Reward Machine decompositions distribute learning difficulty across sequential subtasks. Together, the three research questions highlight the complementary roles of independent RM-state training and end-to-end evaluation. End-to-end experiments reveal which Reward Machine decomposition performs best, whereas independent RM-state training explains why those performance differences

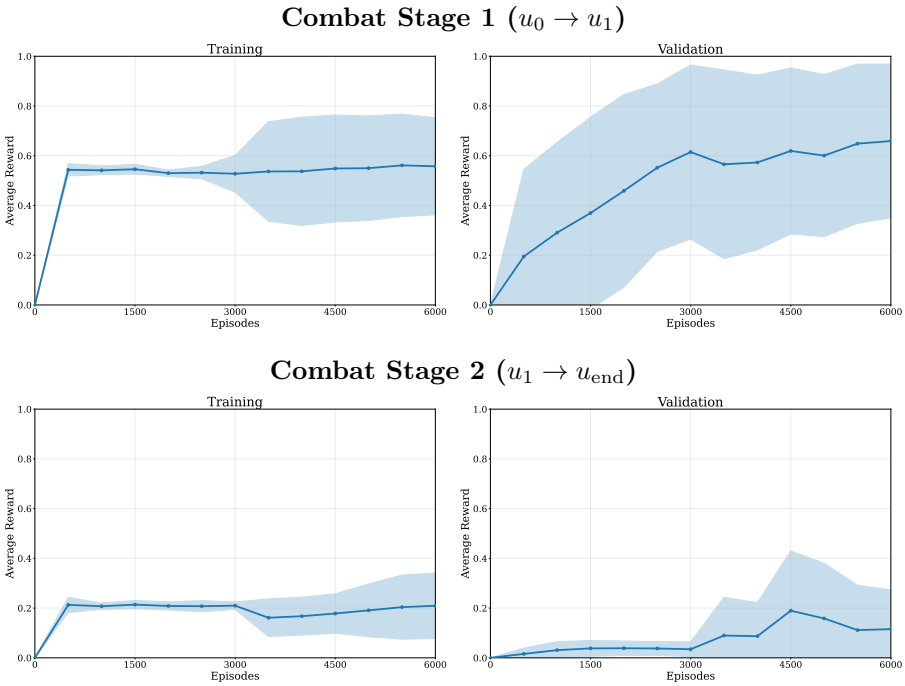


Fig. 10: Learning dynamics for the Reward Machine shown in Fig. 9. Rows correspond to Reward Machine combat stages. Curves report the success rate during independent training of each Reward Machine state using softmax exploration.

arise by exposing how learning difficulty is distributed across individual subtasks. Consequently, the proposed methodology provides a practical tool for analyzing and designing RM before selecting a particular task decomposition.

7 Discussion

In this paper, we investigated how different Reward Machine (RM) decompositions influence RL. Rather than treating the RM as a fixed task description, we treated its decomposition as a design choice. To analyze this, we evaluated both the learning performance of individual RM states and overall task performance. Our results show that two or more RM representing the same task can lead to very different learning behavior. The key factor is not simply the number of intermediate states, but how the decomposition distributes learning difficulty across the task. Adding intermediate objectives does not always make learning easier; it can simply shift where the agent struggles. Evaluating RM states independently reveals these bottlenecks before full end-to-end training. This provides information that final task performance alone cannot capture and offers a practical way to compare alternative Reward Machine decompositions. Overall, the proposed methodology helps explain why different RM representations produce different learning outcomes, making it useful for designing

more effective structured RL tasks. Future work will extend this analysis to larger and automatically generated RM and investigate how the proposed methodology can support their automatic generation and optimization. As recent research has increasingly explored the automatic synthesis of RM, our results suggest that evaluating individual RM states could help identify which of several behaviorally equivalent decompositions is likely to be easier for RL. Finally, the methodology will be evaluated with additional RL algorithms and more challenging environments.

Acknowledgements. Acknowledgments to the Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) for the support provided through the assignment of grants numbered 833132 and 957701.

Declarations

Funding. This work was supported by the Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) through grants numbered 833132 and 957701.

Competing interests. The authors declare no competing interests.

Ethics approval and consent to participate. Not applicable.

Consent for publication. Not applicable.

Data availability. This study used no external datasets. All experimental data were generated through simulation in the publicly available MiniGrid (DoorKey) and VizDoom (Deadly Corridor) environments. The data generated and analyzed during this study (training logs, success rates, and evaluation results) are available from the corresponding author upon reasonable request.

Code availability. The source code used to implement the Reward Machine analysis methodology is available from the corresponding author on reasonable request.

Author contribution. All authors (U.R.-C., M.-A.I.-M., A.O.-G., D.-L.A.-O., and F.F.) contributed equally to this work. U.R.-C. implemented the methods and conducted the experiments; M.-A.I.-M. and D.-L.A.-O. supervised the project; A.O.-G. and F.F. contributed to the methodology and analysis. All authors wrote, reviewed, and approved the final manuscript.

References

[1] Sutton, R.S., Precup, D., Singh, S.: Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence **112**(1–2), 181–211 (1999) https://doi.org/10.1016/S0004-3702(99)00052-1

[2] Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research **13**, 227–303 (2000) https://doi.org/10.1613/jair.639

[3] Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete Event Dynamic Systems **13**(1–2), 41–77 (2003) https://doi.org/10.1023/A:1022940726771

[4] Botvinick, M.M.: Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. Cognition **113**(3), 262–280 (2009) https://doi.org/10.1016/j.cognition.2008.08.008

[5] Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: Using reward machines for high-level task specification and decomposition in reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning (ICML), pp. 2107–2116 (2018)

[6] Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: Reward machines: Exploiting reward function structure in reinforcement learning. Journal of Artificial Intelligence Research **73**, 173–208 (2022) https://doi.org/10.1613/jair.1.13196

[7] Camacho, A., Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: LTL and beyond: Formal languages for reward function specification in reinforcement learning. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), pp. 6065–6073 (2019)

[8] Camacho, A., Varley, J., Zeng, A., Jain, D., Iscen, A., Kalashnikov, D.: Reward machines for vision-based robotic manipulation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 14284–14290 (2021)

[9] Sun, X., Shoukry, Y.: Neurosymbolic motion and task planning for linear temporal logic tasks. IEEE Transactions on Robotics (2024)

[10] Furelos-Blanco, D., Law, M., Jonsson, A., Broda, K., Russo, A.: Hierarchies of reward machines. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 202, pp. 10494–10541 (2023)

[11] Hassanbeig, M., Kroening, D., Neider, D.: Learning reward machines for partially observable reinforcement learning. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 34, pp. 26861–26873 (2021)

[12] Neary, C., Xu, Z., Wu, B., Topcu, U.: Reward machines for cooperative multiagent reinforcement learning. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 934–942 (2021)

[13] Smith, S., Neary, C., Topcu, U.: Automatic decomposition of reward machines for decentralized multiagent reinforcement learning. In: Proceedings of the IEEE

Conference on Decision and Control (CDC), pp. 5423–5430 (2023). https://doi.org/10.1109/CDC49753.2023.10383918

[14] Parr, R., Russell, S.: Reinforcement learning with hierarchies of machines. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 10 (1997)

[15] McGovern, A., Barto, A.G.: Automatic discovery of subgoals in reinforcement learning using diverse density. In: Proceedings of the 18th International Conference on Machine Learning (ICML), pp. 361–368 (2001)

[16] Stolle, M., Precup, D.: Learning options in reinforcement learning. In: Elisseeff, A., Meila, M. (eds.) Abstraction, Reformulation, and Approximation. Lecture Notes in Computer Science, vol. 2371, pp. 212–223. Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45622-8_16

[17] Bacon, P.-L., Harb, J., Precup, D.: The option-critic architecture. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)

[18] Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.B.: Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 29 (2016)

[19] Levy, A., Konidaris, G., Platt, R., Saenko, K.: Hierarchical reinforcement learning with hindsight. In: International Conference on Learning Representations (ICLR) (2019)

[20] Zheng, X., Yu, C., Zhang, M.: Lifelong reinforcement learning with temporal logic formulas and reward machines. Knowledge-Based Systems **257**, 109650 (2022) https://doi.org/10.1016/j.knosys.2022.109650

[21] Parac, R., Nodari, L., Ardon, L., Furelos-Blanco, D., Cerutti, F., Russo, A.: Learning robust reward machines from noisy labels. In: Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR) (2024). https://doi.org/10.24963/kr.2024/85

[22] Ardon, L., Furelos-Blanco, D., Russo, A.: Learning reward machines in cooperative multiagent tasks. In: Amigoni, F., Sinha, A. (eds.) Autonomous Agents and Multiagent Systems: Best and Visionary Papers. Lecture Notes in Computer Science, vol. 14456. Springer, Cham, Switzerland (2024). https://doi.org/10.1007/978-3-031-56255-6_3

[23] Han, S., Dastani, M., Wang, S.: Learning reward structure with subtasks in reinforcement learning. In: Proceedings of the 27th European Conference on Artificial Intelligence (ECAI), pp. 2282–2289 (2024). https://doi.org/10.3233/FAIA240751

[24] Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th International Conference on Machine Learning (ICML),

pp. 41–48 (2009)

[25] Florensa, C., Held, D., Geng, X., Abbeel, P.: Reverse curriculum generation for reinforcement learning. In: Proceedings of the 1st Annual Conference on Robot Learning (CoRL), pp. 482–495 (2017)

[26] Andreas, J., Klein, D., Levine, S.: Modular multitask reinforcement learning with policy sketches. In: Proceedings of the 34th International Conference on Machine Learning (ICML), pp. 166–175 (2017)

[27] Portelas, R., Colas, C., Weng, L., Hofmann, K., Oudeyer, P.-Y.: Automatic curriculum learning for deep RL: A short survey. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), pp. 4819–4825. International Joint Conferences on Artificial Intelligence Organization, ??? (2020). https://doi.org/10.24963/ijcai.2020/671

[28] Littman, M.L., Fu, J., Topcu, G., Kress-Gazit, E., Zilberstein, S.: Environment-Independent Task Specifications via GLTL. arXiv preprint arXiv:1704.04341 (2017)

[29] Li, X., Vasile, C.-I., Belta, C.: Reinforcement learning with temporal logic rewards. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3834–3839 (2017). https://doi.org/10.1109/IROS.2017.8202237

[30] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Cambridge, MA, USA (2018)