

# Decentralizing Multi-Agent Reinforcement Learning with Temporal Causal Information<sup>\*</sup>

Jan Corazza<sup>1</sup> (✉), Hadi Partovi Aria<sup>2</sup>, Hyohun Kim<sup>2</sup>, Daniel Neider<sup>1</sup>, and Zhe Xu<sup>2</sup>

<sup>1</sup> Research Center Trustworthy Data Science and Security of the University Alliance Ruhr, Department of Computer Science, TU Dortmund University, Germany

{jan.corazza,daniel.neider}@tu-dortmund.de

<sup>2</sup> Arizona State University, USA {hpartovi, hkim450, xzhe1}@asu.edu

**Abstract.** Reinforcement learning (RL) algorithms can find an optimal policy for a single agent to accomplish a particular task. However, many real-world problems require multiple agents to collaborate in order to achieve a common goal. For example, a robot executing a task in a warehouse may require the assistance of a drone to retrieve items from high shelves. In Decentralized Multi-Agent RL (DMARL), agents learn independently and then combine their policies at execution time, but often must satisfy constraints on compatibility of local policies to ensure that they can achieve the global task when combined. In this paper, we study how providing high-level symbolic knowledge to agents can help address unique challenges of this setting, such as privacy constraints, communication limitations, and performance concerns. In particular, we extend the formal tools used to check the compatibility of local policies with the team task, making decentralized training with theoretical guarantees usable in more scenarios. Furthermore, we empirically demonstrate that symbolic knowledge about the temporal evolution of events in the environment can significantly expedite the learning process in DMARL.

**Keywords:** Temporal Causality · Multi-Agent Reinforcement Learning · Reward Machines · Formal Methods.

## 1 Introduction

One approach to solving a multi-agent problem is to learn a centralized policy that controls all agents simultaneously. Such a centralized controller is conceptually straightforward, but realizing it is often impractical [6] if the number of agents is large or there is an imposed need for decentralization. As the number of agents increases, the state space grows exponentially, making it less tractable to learn a centralized policy. Decentralization is imposed when agents are physically separated, communication is limited, or privacy is a concern. Centralized policies, by contrast, often assume seamless communication, which is unrealistic in many

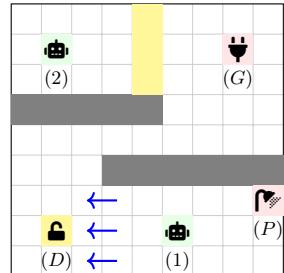
---

\* Code available at <https://github.com/corazza/tcdmarl>

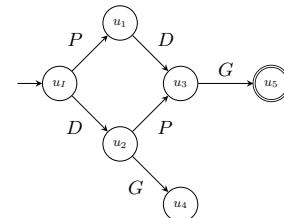
real-world scenarios. This challenge is further exacerbated by sparse reward signals with temporal dependencies, inherent in many real-world tasks.

Instead of learning a centralized policy, agents can learn decentralized policies that allow them to act independently and reduce the need for communication. These independent policies are then combined at execution time to solve the team task. To this end, Neary et al. [7] propose a DMARL algorithm called Decentralized Q-learning with Projected Reward Machines (DQPRM). DQPRM decomposes a global team task specification into a set of local task specifications, one for each individual agent. Agents are trained independently to learn policies based on their local task specifications. To ensure that the agents' local policies, when combined, satisfy the overall team task, DQPRM enforces strict **compatibility criteria** between the local and team specifications. In Section 3, we provide a brief overview of DQPRM.

Reward machines (RMs, formalized in Definition 1) are deterministic, finite-state automata that transduce sequences of relevant high-level events (labels) into sequences of rewards, thereby capturing the temporal nature of a sparse reward signal, and serving as specifications for RL tasks. Reward machines have been widely studied in literature [5,11,3,8,4,2], but the primary focus has been on the single-agent case. In multi-agent problems, communication limitations mean individual agents cannot sense all events, preventing them from accessing the full team state. DQPRM frames this limitation in terms of projected (local) RMs, that capture a particular agent's contribution to the team goal (with respect to events that the agent can sense).



(a) Environment



(b) Team Reward Machine

Fig. 1: Generator Task

To illustrate the limitations of DQPRM's strict compatibility criteria, we will introduce a running example called the *Generator Task*, depicted in Figure 1. In this task, two agents move on a gridworld (Figure 1a), and must collaborate to power a generator during a flooding incident.

Agent 1's task is to prevent the flooding by closing the pipe ( $\text{P}$ ,  $P$ ), and to unlock the door ( $\text{D}$ ,  $D$ ) for agent 2 (in any order). The task for agent 2 is to wait for the door to be unlocked ( $\text{D}$ ), i.e., for the yellow barrier to disappear,

and then power the generator ( $\text{⊕}$ ,  $G$ ). The team reward machine, depicted in Figure 1b, also specifies a failure condition: if agent 2 is let into the room and powers the generator *before* the flooding has been resolved, the team RM will enter an unsafe state,  $u_4$ , from which no further actions can result in a positive reward. The agents obtain a reward of 1 and end the task if their joint actions transition the team RM to into an accepting state,  $u_5$ , and 0 in all steps prior.

The challenge in this example arises from the agents’ limited communication about the task’s status. In particular, agent 1 does not communicate whether it is safe to power up the generator (i.e., whether event  $\text{F}^\triangleright$  has happened). In other words, agent 2 does not know whether the team is collectively in state  $u_2$  ( $\text{⊕}$  is unsafe) or  $u_3$  ( $\text{⊕}$  is safe and completes the task). Although both agents have an individual strategy that solves the team task, this specification fails to meet the requirements of DQPRM, which strictly prohibit outcomes to depend on local event synchronization.

Recent work in causal reinforcement learning [8,3] proposes methods that leverage causal knowledge of the long-term temporal evolution of high-level environmental events to improve the exploration-exploitation trade-off. Paliwal et al. [8] introduce Temporal Logic-based Causal Diagrams (TL-CDs), a formalism that captures such *temporal-causal* knowledge by enabling experts to specify causal relationships between  $LTL_f$  formulas. During learning, temporal-causal knowledge expressed in TL-CDs is used to predict whether further exploration within a given episode is necessary and, if not, to short-circuit the exploration process. Unfortunately, this method lacks rigorous theoretical guarantees. Corazza et al. [3] extend this line of work with a provably correct method that directly integrates temporal-causal knowledge from TL-CDs into the reward machine. Ultimately, both methods expedite convergence to optimal policies by exploiting knowledge of temporal causality—i.e., the long-term evolution of high-level events within the MDP. In Section 4, we provide a brief overview of TL-CDs and their integration with reward machines.

Our paper connects these two lines of work (decentralized MARL and temporal-causal RL). In Section 5, we propose a method to exploit temporal-causal knowledge about the task at hand in order to (1) relax the theoretical compatibility criteria of DQPRM, broadening the scope of multi-agent tasks that can be decomposed and learned in a decentralized manner; and (2) short-circuit certain explorations during decentralized learning, improving the sample efficiency of the learning process even further. Moreover, we prove that our relaxed compatibility criterion is consistent with the original one, in the sense that passing the original criterion implies passing the relaxed one. In Section 6, we empirically demonstrate our method’s effectiveness on the *Generator Task* and introduce the *Laboratory Task*, presenting experimental results for both. An analysis of a third case study, the *Buttons Task*, is provided in Appendix C.

## 2 Preliminaries

Aside from acting as a task specification, reward machines provide a form of finite memory for a team or an agent, tracking progress through the task. We assume that during the agents’ interaction with the MDP, relevant high-level events are labeled. Reward machines use these event labels as inputs to transition between states and determine the appropriate rewards. By modeling the temporal dependencies within the task, RMs guide agents through complex environments, especially in situations where rewards are sparse. We first introduce the general definitions of Event-based Reward Machines (RMs) and RM-based Markov Decision Processes (RM-MDPs), which provide a foundational framework for handling task specification and reward functions. In Section 3, we will extend this formalism to the multi-agent setting, where decentralized coordination becomes crucial.

**Definition 1 (Event-based Reward Machine).** *An event-based RM  $\mathcal{R} = \langle U, u_I, \Sigma, \delta, \sigma, F \rangle$  is a tuple where  $U$  is a finite set of states with an initial state  $u_I \in U$ ,  $\Sigma$  is the set of events,  $\delta : U \times \Sigma \rightarrow U$  is a (partial) transition function,  $\sigma : U \times U \rightarrow \mathbb{R}$  is a (partial) function mapping transitions to rewards, and  $F \subseteq U$  is a set of terminal states that signal the end of the interaction.*

When the RM is in state  $u \in U$  and reads an event  $e \in \Sigma$  such that  $(u, e) \in \text{Dom}(\delta)$ , it transitions into a new state  $u' = \delta(u, e)$  and outputs a reward  $\sigma(u, u')$ . Otherwise, the RM remains in the same state and outputs a reward of 0. For finite event sequences, we define  $\delta(u, \epsilon) = u$  and  $\delta(u, \xi e) = \delta(\delta(u, \xi), e)$ , where  $\epsilon$  is the empty sequence,  $\xi \in (\Sigma)^*$ , and  $e \in \Sigma$ . Note that  $\delta$  and  $\sigma$  are partial functions, i.e., we do not require the transition function to be defined for every input event from every state. The term *event-based RM* is a deliberate departure from prior work [7], which conflated event-driven RMs with propositional RMs under the generic “RM” label. We focus on *task completion* RMs which output a reward of 1 upon reaching a terminal state  $u \in F$ , and 0 otherwise. For brevity, we refer to task completion event-based reward machines simply as reward machines throughout the remainder of this work.

To connect an RM with the MDP (with set of states  $S$ ), we use a labeling function  $L : S \times U \rightarrow 2^\Sigma$ . The codomain of  $L$  is  $2^\Sigma$  to model the possibility that certain events may occur concurrently. We restrict the labeling function so that only one event may occur per step, per agent.<sup>3</sup> Because of this restriction and the compatibility criterion defined in Section 3, the order an RM reads simultaneous events does not matter. In other words, if an RM reads an event sequence  $\xi \in \text{Perm}(L(s, u, s'))$ , the resulting state  $u' = \delta(u, \xi)$  is well-defined (independent of the permutation of  $\xi$ ).

**Definition 2 (RM-MDP).** *A Reward Machine-based Markov Decision Process is a tuple  $\mathcal{M} = \langle S, s_I, A, p, \gamma, \mathcal{R}, L \rangle$  consisting of a finite state space  $S$ , an*

---

<sup>3</sup> We explain this assumption in Section 3, where we provide background for decentralized MARL.

initial state  $s_I \in S$ , a finite set of actions  $A$ , a probabilistic transition function  $p: S \times U \times A \times S \rightarrow [0, 1]$ , a discount factor  $\gamma \in (0, 1]$ , a reward machine  $\mathcal{R}$  which captures the reward function, and a labeling function  $L: S \times U \times S \rightarrow 2^\Sigma$ . Note that the transition probabilities  $p(s' | s, u, a)$  are conditioned on the state of the reward machine  $u \in U$ .

Intuitively, if the MDP and the RM are in a joint state  $(s, u) \in S \times U$ , and the agent chooses an action  $a \in A$ , the MDP transitions to a new state  $s' \sim p(s, u, a)$ . The labeling function then outputs a set of simultaneous events  $L(s, u, s')$ , which the RM reads in any order, and transitions into  $u' = \delta(u, L(s, u, s'))$ .

We say that a trajectory  $s_0 u_0 a_0 \cdots a_{n-1} s_n u_n$  is attainable within an RM-MDP  $\mathcal{M} = \langle S, s_I, A, p, \gamma, \mathcal{R}, L \rangle$  when  $(s_0, u_0) = (s_I, u_I)$ , and for all  $i = 0, 1, \dots, n-1$  we have  $p(s_i, u_i, a_i, s_{i+1}) > 0$  and  $u_{i+1} = \delta(u_i, L(s_i, u_i, s_{i+1}))$ . An event sequence  $\xi = e_1 \dots e_n$  is attainable if it can be obtained using a finite number of concatenation and swap operations from the label sequence  $\ell_i = L(s_i, u_i, s_{i+1})$  of an attainable trace.

The reward function of an RM-MDP is induced by the reward machine  $\mathcal{R}$  via  $R((s, u), a, (s', u')) = \sigma(u, u')$ . Because  $R$  is Markovian over the product space  $S \times U$ , one may use Q-learning to find an optimal policy in an RM-MDP.

### 3 Decentralized Multi-Agent Reinforcement Learning

In this section, we outline the baseline DQPRM algorithm proposed by Neary et al. [7], including its assumptions and theoretical guarantees. In decentralized MARL with reward machines, the overall team task is specified by an RM  $\mathcal{R}$  defined over a global set of events  $\Sigma$  (all events that may occur in the MDP). However, not every agent can observe all events from  $\Sigma$ . In order to capture this limitation, we define a local event set  $\Sigma_i$  for each agent  $i = 1, \dots, N$ , where  $N$  is the number of agents. We assume that  $\Sigma = \bigcup_{i=1}^N \Sigma_i$ .

In the *Generator Task* from Figure 1, the local event set of agent 1 is  $\Sigma_1 = \{P, D\}$ , modeling the capabilities to fix the pipe (observe and open the door (observe ). The local event set of agent 2 is  $\Sigma_2 = \{D, G\}$ , modeling the capabilities to observe the door being unlocked () and power the generator (). As demonstrated by the event  $D$ , local event sets may overlap, i.e., there may be events that can be sensed by more than one agent. We call such events *shared events*, and write  $I_e = \{i \mid e \in \Sigma_i\}$  for the set of agents that share event  $e$ . When  $|I_e| > 1$  for some  $e \in \Sigma$ , a synchronization mechanism ought to be simulated during decentralized training episodes. For example, since the event  $D$  is under the control of agent 1, during decentralized training of agent 2, we simulate the communication of this event with probability  $p > 0$  at every time-step.

In decentralized MARL, we aim to train each agent independently, so as to avoid learning a centralized policy. The primary benefit of this approach is sample efficiency, as centralized policies induce a state space that grows exponentially with the number of agents. Other benefits include privacy (limiting shared knowledge), and the fact that centralized policies may be infeasible due to practical concerns (differing capabilities, limited communication).

To facilitate decentralized training, we first need to capture the individual contribution of each agent towards the team goal in terms of the agent's local event set. This can be achieved with the notion of *Projected Reward Machines*, which we will now introduce. We also refer to projected reward machines as *local RMs*, to highlight that the projection is defined in terms of the local event set.

To define the projection of the team RM  $\mathcal{R} = \langle U, u_I, \Sigma, \delta, \sigma, F \rangle$  along a local event set  $\Sigma_i \subseteq \Sigma$ , we first introduce an equivalence relation  $\sim_i$  over states of  $\mathcal{R}$ . Informally, two states  $u_1, u_2 \in U$  are  $\sim_i$ -equivalent if agent  $i$  can not distinguish between them based on events in  $\Sigma_i$ . Formally,  $\sim_i$  is the smallest equivalence relation such that for all  $u_1, u_2, u'_1, u'_2 \in U$  we have (1)  $u_1 \sim_i u_2$  if there exists an event  $e \in \Sigma \setminus \Sigma_i$  such that  $u_2 = \delta(u_1, e)$ ; and (2) if  $u_1 \sim_i u'_1$  and there exists an event  $e \in \Sigma_i$  such that  $\delta(u_1, e) = u_2$  and  $\delta(u'_1, e) = u'_2$ , then  $u_2 \sim_i u'_2$ . We denote the set of equivalence classes of  $\sim_i$  with  $U / \sim_i$ , and we will write  $[u]_i$  for the  $\sim_i$ -equivalence class of  $u \in U$ .

Condition (1) ensures that two states of  $\mathcal{R}$  are indistinguishable for agent  $i$  if a transition between them is triggered by an event outside of  $\Sigma_i$ . Condition (2) (congruence) ensures that an event  $e \in \Sigma_i$  may only trigger transitions to a unique successor equivalence class. An algorithm to compute this equivalence relation with runtime  $O(|U|^7|\Sigma_i|^2 + |U|^5|\Sigma_i| + |U|^4)$  is provided in appendix C of Wong [10]. Using  $\sim_i$ , we formalize the notion of a projected RM in Definition 3. Figure 2 depicts the projections of the team RM from the *Generator Task* along the local event sets of agents 1 and 2.

**Definition 3 (Projected Reward Machine).** Given a reward machine  $\mathcal{R} = \langle U, u_I, \Sigma, \delta, \sigma, F \rangle$  and local event set  $\Sigma_i$ , the projection of  $\mathcal{R}$  along  $\Sigma_i$  is the RM  $\mathcal{R}_i = \langle U_i, u_I^i, \delta_i, \Sigma_i, F_i \rangle$  where  $U_i = U / \sim_i$ ,  $u_I^i = [u_I]_i$ ,  $\delta_i : U_i \times \Sigma_i \rightarrow U_i$  is defined as  $u_2^i = \delta_i(u_1^i, e)$  for  $e \in \Sigma_i$  if and only if there exist  $u_1, u_2 \in U$  such that  $[u_1] = u_1^i$ ,  $[u_2] = u_2^i$  and  $u_2 = \delta(u_1, e)$  (and undefined otherwise),  $F_i = \{u^i \in U_i \mid \exists u \in F, u^i = [u]_i\}$ , and  $\sigma_i(u_1^i, u_2^i) = \mathbf{1}_{u_2^i \in F_i \wedge u_1^i \notin F_i}$ .

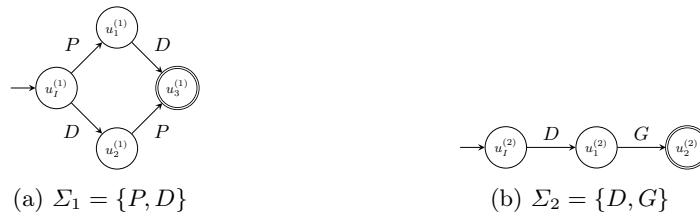


Fig. 2: Projections of the RM from Figure 1b along local event sets of agents 1 (Left) and 2 (Right).

We model a multi-agent system with  $N$  agents as an RM-MDP  $\mathcal{M} = \langle S, s_I, A^N, p, \gamma, \mathcal{R}, L \rangle$  where the state space  $S$  is a Cartesian product of local state spaces,  $S = S_1 \times \dots \times S_N$ , the action space  $A^N$  is a product of individual action spaces (the same for each agent)  $A \times \dots \times A$ , and the transition

function  $p : S \times U \times A^N \times S$  can be decomposed as a product  $p(s, u, a, s') = \prod_{i=1}^N p_i(s_i, u^i, a_i, s'_i)$ , where  $u^i = [u]_i$  is the projection of the team state from  $\mathcal{R}$  to the local RM state of agent  $i$ . The dynamics  $p_i$  of agent  $i$  depend only on the agent's individual state and action,  $(s_i, a_i)$ , and the events that the agent can sense (i.e., the state of its local RM  $\mathcal{R}_i$ ).

To fully capture the local dynamics of each agent, DQPRM relies on the notion of local labeling functions  $L_i : S_i \times U_i \times S_i \rightarrow 2^{\Sigma}$ , where  $S_i$  is the local state space of agent  $i$ , and  $U_i = U / \sim_i$  are the states of its local RM. Intuitively,  $L_i(s_i, u^i, s'_i)$  outputs events that *could* be occurring from the perspective of agent  $i$ , given that the rest of the team state is unknown. We note that local labeling functions must output at most a single event  $\{e\}$  (which corresponds to the assumption that only one event may occur per agent per time step), and that  $L(s, u, s')$  outputs event  $e$  if and only if  $L_i(s_i, u^i, s'_i)$  output  $e$ , for all  $i \in I_e$ . We call such label functions decomposable with respect to  $\Sigma_1, \dots, \Sigma_N$  (or just decomposable if the family  $\{\Sigma_i\}_{i=1}^N$  is clear from the context).<sup>4</sup> We define the projection of a finite event sequence  $\xi \in \Sigma^*$  onto the local event set  $\Sigma_i$  as  $P_i(\xi' \ell) = P_i(\xi')e$  if  $\xi = \xi'e$  and  $e \in \Sigma_i$ , and  $P_i(\xi e) = P_i(\xi)$  otherwise (for the empty word  $\epsilon$ ,  $P_i(\epsilon) = \epsilon$  for all  $i = 1, \dots, N$ ).

A key requirement for the task decomposition to be successful is that policies induced by projected RMs must combine to form a policy that satisfies the team RM. This requirement is satisfied if the team RM is bisimilar to the parallel composition of its projections, which we refer to as the *strict decomposition criterion*. Intuitively, two reward machines  $\mathcal{R}$  and  $\mathcal{P}$  are bisimilar ( $\mathcal{R} \cong \mathcal{P}$ ) if there exists a relation between their states such that the transition behavior of related states is the same (initial and final states must be related). The parallel composition of two RMs  $\mathcal{R}_1$  and  $\mathcal{R}_2$  is an RM  $\mathcal{R}_1 \parallel \mathcal{R}_2$  whose states are the Cartesian product of states of  $\mathcal{R}$  and  $\mathcal{P}$ , initial (terminal) states are pairs of initial (terminal) states, and transitions are defined component-wise.<sup>5</sup> Using the notions of bisimilarity and parallel composition of RMs, we can formalize the strict decomposition criterion for the team RM  $\mathcal{R}$  and local RMs  $\mathcal{R}_1, \dots, \mathcal{R}_N$ . The criterion holds if  $\mathcal{R} \cong \parallel_{i=1}^N \mathcal{R}_i$ . In that case, decentralized training of local policies will yield a combined policy with guaranteed bounds on the probability of solving the team task. By definition, if local RMs  $\mathcal{R}_i$  output 1 (for all  $i = 1, \dots, N$ ), then their parallel composition also outputs 1. And because their parallel composition is bisimilar to the team RM, the team task is satisfied as well. The converse also holds: if the team RM outputs a reward, then each local RM will output a reward. This is formalized in Theorem 1 [7].

**Theorem 1 (Strict Decomposition Criterion).** *If RM  $\mathcal{R}$  and projections  $\mathcal{R}_1, \dots, \mathcal{R}_N$  satisfy the strict decomposition criterion, then  $\forall \xi \in \Sigma^*, \mathcal{R}(\xi) = 1$  if and only if  $\mathcal{R}_i(P_i(\xi)) = 1, \forall i = 1, \dots, N$ . Here,  $P_i(\xi)$  is the projection of  $\xi$  to  $\Sigma_i$  defined earlier.*

<sup>4</sup> For more details on constructing local labeling functions see Neary et al. [7]

<sup>5</sup> Formal definitions in Appendix A. Visual explanation in Appendix B.2.

We provide the pseudocode for DQPRM in Algorithm 1. In brief, DQPRM runs  $N$  concurrent episodes, where each agent moves independently, receiving observations, events, and rewards decoupled from the state of the other  $N - 1$  agents. When acting in a team setting during testing, agents must synchronize on shared events. This is the only communication channel necessary to execute single-agent policies, found by DQPRM, in a multi-agent RM-MDP. When the local labeling function  $L_i$  outputs a shared event  $e$ , synchronization ensures that  $\mathcal{R}_i$  reads  $e$  iff.  $L_j$  outputs the same event  $e$  for all  $j \in I_e$ . To account for this fact during decentralized training, DQPRM simulates the synchronization signal with probability  $p > 0$  (Line 18). We summarize the guarantees for team performance in Theorem 2.

---

**Algorithm 1:** Decentralized training with DQPRM

---

**Input:** Team RM  $\mathcal{R}$ , local event sets  $\Sigma_1, \dots, \Sigma_N$ , local labeling functions  $L_1, \dots, L_N$

**Output:** Policies  $\pi_1, \dots, \pi_N$

- 1 Project  $\mathcal{R}$  along local event sets  $\Sigma_i$  to obtain projected RMs  $\mathcal{R}_1, \dots, \mathcal{R}_N$ ;
- 2 **if**  $\mathcal{R} \not\cong \mathcal{R}_1 \parallel \dots \parallel \mathcal{R}_N$  **then**
- 3 | Reject: the strict decomposition criterion does not hold;
- 4 **end**
- 5 Initialize( $\pi_1, \dots, \pi_N$ );
- 6 **for**  $m = 1$  **to**  $numEpisodes$  **do**
- 7 | **for**  $i = 1$  **to**  $N$  **do**
- 8 | |  $s_i \leftarrow s_I^i$ ,  $u_i \leftarrow u_I^i$ ;
- 9 | **end**
- 10 | **for**  $t = 1$  **to**  $numSteps$  **do**
- 11 | | **for**  $i = 1$  **to**  $N$  **do**
- 12 | | | **if**  $TaskComplete_i(u_i)$  **then**
- 13 | | | | **continue**;
- 14 | | | **end**
- 15 | | |  $a_i \leftarrow \text{Sample}(\pi_i(s_i, u_i))$ ;
- 16 | | |  $s' \leftarrow \text{Step}_i(s_i, u_i, a_i)$ ,  $u' \leftarrow u_i$ ,  $r_i \leftarrow 0$ ;
- 17 | | |  $// L_i(s_i, u_i, s')$  is either  $\emptyset$  or  $\{e\} \in \Sigma_i$
- 18 | | | **for**  $e \in L_i(s_i, u_i, s')$  **do**
- 19 | | | | **if**  $|I_e| = 1$  **or**  $\text{Rand}(0, 1) \leq p$  **then**
- 20 | | | | |  $u' \leftarrow \delta(u_i, e)$ ;  $r_i \leftarrow r_i + \sigma(u_i, u')$ ;  $u_i \leftarrow u'$ ;
- 21 | | | | **end**
- 22 | | | **end**
- 23 | | |  $\text{PolicyUpdate}(\pi_i, s_i, u_i, a_i, s', r_i)$ ;
- 24 | | |  $s_i \leftarrow s'$ ;
- 25 | **end**
- 26 | **end**
- 27 **return**  $(\pi_1, \dots, \pi_N)$ ;

---

**Theorem 2 (Decomposition Viability).** *Given a team RM  $\mathcal{R}$ , local event sets  $\Sigma_1, \dots, \Sigma_N$ , a decomposable label function  $L$ , and local labeling functions  $L_1, \dots, L_N$ , assume that agents synchronize on shared events and  $\mathcal{R} \cong \parallel_{i=1}^N \mathcal{R}_i$ . Then for all team trajectories  $s_0 u_0 \cdots s_k u_k$  and local trajectories  $\{s_0^i u_0^i \cdots s_k^i u_k^i\}_{i=1}^N$ , we have  $\mathcal{R}(L(s_0 u_0 \cdots s_k u_k)) = 1$  if and only if  $\mathcal{R}_i(L_i(s_0^i u_0^i \cdots s_k^i u_k^i)) = 1$  for all  $i = 1, \dots, N$ . Furthermore, let  $V^\pi(s_I)$  denote the team success probability and  $V_i^\pi(s_I)$  the success probability of agent  $i$ . Then  $\max\{0, V_1^\pi(s_I) + \cdots + V_N^\pi(s_I) - (N-1)\} \leq V^\pi(s_I) \leq \min\{V_1^\pi(s_I), \dots, V_N^\pi(s_I)\}$ .*

Formal proofs of Theorems 1 and 2 can be found in Neary et al. [7]. Informally, Theorem 2 states that if a task specification respects the strict decomposition criterion, decentralized training will yield local agent policies which, when combined, satisfy the team task (depending on the probability that each agent performs its own part of the task). This result depends critically on Theorem 1, which guarantees a correspondence between solving local tasks and solving the team task, and the assumption that the agents synchronize on shared events.

## 4 Modeling Temporal-Causal Knowledge with TL-CDs

In this section, we lay the groundwork for modeling temporal-causal knowledge in RM-MDPs, and extend the notion of Temporal Logic-based Causal Diagrams (TL-CDs), proposed in Paliwal et al. [8], to Event-based Reward Machines. Linear temporal logic over finite sequences ( $LTL_f$ ) is a formal reasoning system that can capture causal and temporal properties of event sequences and RM-MDPs. Aside from Boolean operators like  $\neg$  and  $\vee$ ,  $LTL_f$  introduces temporal operators such as  $\mathbf{G}\psi$  (true if and only if  $\psi$  holds for every element in the sequence),  $\mathbf{X}\psi$  (true iff.  $\psi$  holds for the next element of the sequence), and  $\psi \mathbf{U} \varphi$  (true iff.  $\psi$  holds until  $\varphi$  becomes true, and  $\varphi$  is true in some element of the sequence). We also rely on the weak until operator  $\psi \mathbf{W} \varphi$  (true iff.  $\psi$  holds until  $\varphi$  becomes true, but  $\varphi$  is not required to become true).

TL-CDs are directed graphs where nodes are labeled with  $LTL_f$  formulas and edges represent causal relationships between them, providing a structured notation to express temporal-causal knowledge. To give the semantics of TL-CD  $\mathcal{C}$ , one can construct an equivalent  $LTL_f$  formula,  $\varphi^{\mathcal{C}}$ , which captures the temporal-causal knowledge encoded in the graph. The formula induced by the TL-CD in Figure 3a,  $\mathbf{G}(D \rightarrow \mathbf{G}(\neg X P))$ , models sequences in which the event  $P$  never follows the event  $D$ . In other words, it captures the effect of the one-way ramp (represented by blue arrows  $\leftarrow$  in Figure 1a), which blocks agent 1 from returning to the region of the MDP containing the pipe (☒) after unlocking the door (🔒).

To capture the semantics of a given TL-CD  $\mathcal{C}$  over event sequences in  $\Sigma^*$ , we construct an equivalent  $LTL_f$  formula,  $\Psi_{\Sigma}^{\mathcal{C}}$ , as the conjunction  $\Psi_{\Sigma}^{\mathcal{C}} \equiv \varphi^{\mathcal{C}} \wedge \psi_{\Sigma}$ . The formula  $\varphi^{\mathcal{C}}$  encodes the temporal-causal knowledge expressed in  $\mathcal{C}$ , and is defined as  $\varphi^{\mathcal{C}} \equiv \bigwedge_{\varphi \blacktriangleright \psi} \mathbf{G}(\varphi \rightarrow \psi)$ , where  $\varphi \blacktriangleright \psi$  iterates over edges that connect formulas  $\varphi$  and  $\psi$  in the TL-CD. The formula  $\psi_{\Sigma} = \mathbf{G}(\bigvee_{e \in \Sigma} (e_i \wedge (\bigwedge_{e' \in \Sigma \setminus \{e\}} \neg e')))$



Fig. 3: TL-CD for the Generator Task (Left) and respective Causal DFA (Right)

restricts models of  $\Psi_{\Sigma}^C$  to event sequences in  $\Sigma^*$ . If  $\Psi_{\Sigma}^C$  is true for an event sequence  $\xi = e_1 \dots e_n$ , we will write  $\xi \models \Psi_{\Sigma}^C$ . We will say that a TL-CD  $C$  holds for an MDP  $\mathcal{M}$  if for every event sequence  $\xi$  attainable in  $\mathcal{M}$ , we have  $\xi \models \Psi_{\Sigma}^C$ . In order to simplify working with TL-CDs, we leverage the notion of deterministic finite automata (DFAs), formalized in Definition 4.

**Definition 4 (Deterministic Finite Automaton).** A DFA is a tuple  $C = (Q, q_I, \Sigma, \delta, F)$  consisting of a finite set of states  $Q$  with an initial state  $q_I$ , input alphabet  $\Sigma$ , deterministic transition function  $\delta : Q \times \Sigma \rightarrow Q$ , and a set of accepting states  $F \subseteq Q$ .

If the run of the DFA  $C$  on an input string  $\xi$  ends in an accepting state  $q \in F$ , we will write  $\xi \in \mathcal{L}(C)$ . Every TL-CD  $C$  can be converted into an equivalent DFA  $C$  [8], in the sense that for every  $\xi$ , we have  $\xi \in \mathcal{L}(C) \iff \xi \models \Psi_{\Sigma}^C$ . We will refer to  $C$  as the *causal DFA*. When illustrating causal DFAs, we will only consider the fragment constructed from  $\varphi^C$ , as the contribution of  $\psi_{\Sigma}$  is always the same and serves a technical purpose. One can obtain the full causal DFA from the parallel composition of automata for  $\varphi^C$  and  $\psi_{\Sigma}$ . In Figure 3b, we illustrate the DFA induced by the TL-CD which holds for the *Generator Task*.

## 5 Causal DQPRM

We enhance decentralized multi-agent RL by integrating temporal-causal knowledge through TL-CDs in two key ways. *First*, we enable valid decomposition of tasks rejected by DQPRM through causal constraints, preserving performance guarantees. *Second*, we accelerate learning by short-circuiting redundant exploration using TL-CD predictions, significantly improving sample efficiency in decentralized training.

### 5.1 Relaxing the Decomposition Criterion in DQPRM

The main assumption of the method proposed in Neary et al. [7] is that the team RM is bisimilar to the parallel composition of the projected RMs. As demonstrated by the decomposition in Figure 2, this assumption is not always satisfied. To illustrate this point, consider the event sequence  $\xi = DGP$ , which induces a reward of 1 in both projections, but 0 in the team RM. Unfortunately, this means that Theorem 2 no longer guarantees a viable decomposition into

local RMs. As such, one can not expect DQPRM to solve the *Generator Task* (at least not before results analogous to Theorem 1 are shown to hold).

On the other hand, by constructing a strategy for each agent manually, one can see that decentralized learning ought to yield satisfactory results. The reason for this lies in the fact that  $\text{P}^\times(P)$  and  $\text{L}^\times(D)$  are separated by a one-way ramp (Figure 1a). Once agent 1 visits  $\text{L}^\times$ , then due to this one-way ramp, no attainable trajectory leads to  $\text{P}^\times$ . In other words, while agent 1 satisfies its local task (Figure 2a) by observing  $P$  and  $D$  in any order, the structure of the MDP does not permit orders of the form  $P \rightarrow \dots \rightarrow D$ . Therefore, Q-learning, equipped with knowledge of the current RM state of agent 1, ought to converge to a policy which avoids such paths.

As TL-CDs overapproximate the set of attainable event sequences within the MDP, they enable experts to encode temporal-causal knowledge—for instance, the fact that problematic sequences like  $\xi = DGP$  are unattainable. Given a team task RM  $\mathcal{R}$  and local event sets  $\Sigma_1, \dots, \Sigma_N$  where  $\mathcal{R} \not\cong \parallel_{i=1}^N \mathcal{R}_i$ , our method leverages TL-CD  $\mathcal{C}$  to validate whether decomposition remains viable under the causal constraints of the RM-MDP.

To relax the strict bisimilarity assumption ( $\mathcal{R} \cong \parallel_{i=1}^N \mathcal{R}_i$ ) from Theorem 1, we introduce a modified bisimulation check:  $\mathcal{R}\|\mathcal{C} \cong \mathcal{P}\|\mathcal{C}$ , where  $\mathcal{P} = \parallel_{i=1}^N \mathcal{R}_i$  represents the parallel composition of projected RMs, and  $\mathcal{R}\|\mathcal{C}$ ,  $\mathcal{P}\|\mathcal{C}$  denote their respective compositions with the causal DFA  $\mathcal{C}$  derived from  $\mathcal{C}$ .

The parallel composition of an RM and a causal DFA synchronizes their transitions, meaning that  $\mathcal{R}\|\mathcal{C}$  and  $\mathcal{P}\|\mathcal{C}$  do not reward event sequences which do not satisfy the TL-CD  $\mathcal{C}$ . In Theorem 3 we show that our relaxed decomposition criterion yields the same guarantees as the strict one, and in Theorem 4 we show that the two are compatible (provided  $\mathcal{C}$  holds for the RM-MDP).

As  $\mathcal{R}$  and  $\mathcal{P}$  are both deterministic finite automata, if  $\mathcal{R} \not\cong \mathcal{P}$ , the issue is caused by event sequences  $\xi$  such that  $\mathcal{R}(\xi) \neq \mathcal{P}(\xi)$  [1]. At its core, our method allows experts to exploit temporal-causal knowledge by finding a TL-CD  $\mathcal{C}$  such that (1)  $\mathcal{C}$  holds for the RM-MDP, i.e., for every event sequence  $\xi$  attainable in  $\mathcal{M}$ , we have  $\xi \models \Psi_\Sigma^\mathcal{C}$ ; and (2)  $\mathcal{R}$ ,  $\mathcal{P}$ , and the causal DFA  $\mathcal{C}$  of the TL-CD  $\mathcal{C}$  pass the relaxed decomposition criterion, i.e.  $\mathcal{R}\|\mathcal{C} \cong \mathcal{P}\|\mathcal{C}$ .

The first property ensures that the TL-CD overapproximates the RM-MDP and expresses correct causal knowledge about the environment. The second property ensures that the causal knowledge added by the TL-CD is complete, in the sense that for every event sequence  $\xi$  such that  $\mathcal{P}(\xi) \neq \mathcal{R}(\xi)$ , we have  $\xi \not\models \Psi_\Sigma^\mathcal{C}$  (i.e.  $\xi$  is not an attainable event sequence in the RM-MDP). We use these properties to recover the important result that underpins team performance guarantees, and formalize our relaxed decomposition criterion in Theorem 3. In Theorem 4, we show that the relaxed and strict decomposition criteria are compatible, and in Theorem 5 we provide a lower and upper bound on the team success probability, analogous to Theorem 2. The proofs for the theorems in this section are provided in Appendix B.

**Theorem 3 (Relaxed Decomposition Criterion).** *Let  $\mathcal{R}$  be the team task RM, and  $\mathcal{R}_1, \dots, \mathcal{R}_N$  be the projections of  $\mathcal{R}$  onto the local event sets  $\Sigma_1, \dots, \Sigma_N$ ,*

respectively. If  $\mathcal{C}$  is a TL-CD with causal DFA  $\mathsf{C}$  such that  $\mathcal{R} \parallel \mathsf{C} \cong (\|_{i=1}^N \mathcal{R}_i) \parallel \mathsf{C}$ , then for all  $\mathcal{M}$ -attainable event sequences  $\xi$ , we have  $\mathcal{R}(\xi) = 1$  if and only if  $\mathcal{R}_i(P_i(\xi)) = 1$  ( $\forall i = 1, \dots, N$ ). Proof in Appendix B.1.

Theorem 3 states that if one can find a TL-CD  $\mathcal{C}$  which satisfies the two properties outlined above, then the team task RM can be decomposed into local task RMs such that on all attainable sequences, the team RM and the parallel composition of local RMs output the same reward. Intuitively, if agents 1 through  $N$  satisfy their local tasks (induce a reward of 1 in projected RMs), then their combined behavior also induces a reward of 1 in the parallel composition of projected RMs (which is bisimilar to the team RM).

If the strict decomposition criterion holds, it is not immediately clear that the relaxed criterion will also hold (given an arbitrary TL-CD that holds for the MDP). In other words, Theorem 1 and Theorem 3 present two different criteria for decomposing the team task RM (C1 and C2, respectively), and it is natural to ask whether these criteria are compatible. Table 1 summarizes the possible outcomes of comparing the two criteria.

C1	C2	Explanation
✗	✗	No decomposition (neither met)
✗	✓	New decomposition (our method)
✓	✗	Claim: Impossible combination
✓	✓	Known decomposition (both met)

Table 1: Results from Theorems 3 & 4

**Theorem 4 (Criterion Compatibility).** *Given a team task RM  $\mathcal{R}$ , a family of local event sets  $\Sigma_1, \dots, \Sigma_N$ , and a TL-CD  $\mathcal{C}$  with causal DFA  $\mathsf{C}$ , let  $\mathcal{P} = \|_{i=1}^N \mathcal{R}_i$  be the parallel composition of local task DFAs. Then  $\mathcal{R} \cong \mathcal{P} \Rightarrow \mathcal{R} \parallel \mathsf{C} \cong \mathcal{P} \parallel \mathsf{C}$ . In other words, the additional parallel composition with the causal DFA will not change the bisimulation decision if the original parallel composition is bisimilar to the team task DFA. Proof in Appendix B.2.*

In Theorem 4 we show that a parallel composition with an appropriate causal DFA introduced by our method will not make a task RM, that was originally decomposable, *not* decomposable. In other words, the relaxed criterion is a generalization of the original criterion, and so the two criteria are compatible. Once the relationship between the two criteria has been established, the main result is given by Theorem 5, which provides lower and upper bounds on the probability of combined action success, analogous to Theorem 2. In order to derive the results, we also assume that agents synchronize on shared events in the team setting, and that the team labeling function  $L$  is decomposable with respect to the local event sets with corresponding labeling functions  $L_i$ .

**Theorem 5 (Relaxed Decomposition Viability).** *Given a team RM  $\mathcal{R}$ , local event sets  $\Sigma_1, \dots, \Sigma_N$ , a decomposable label function  $L$ , and local labeling functions  $L_1, \dots, L_N$ , assume that agents synchronize on shared events and  $\mathcal{R}||\mathcal{C} \cong (\|_{i=1}^N \mathcal{R}_i) || \mathcal{C}$ . Then for all team trajectories  $s_0 u_0 \cdots s_k u_k$  and local trajectories  $\{s_0^i u_0^i \cdots s_k^i u_k^i\}_{i=1}^N$ , we have  $\mathcal{R}(L(s_0 u_0 \cdots s_k u_k)) = 1$  if and only if  $\mathcal{R}_i(L_i(s_0^i u_0^i \cdots s_k^i u_k^i)) = 1$  for all  $i = 1, \dots, N$ . Furthermore, we retain the bounds for the team success probability  $V^\pi(s_I)$ , i.e.  $\max\{0, V_1^\pi(s_I) + \cdots + V_N^\pi(s_I) - (N-1)\} \leq V^\pi(s_I) \leq \min\{V_1^\pi(s_I), \dots, V_N^\pi(s_I)\}$ . Proof in Appendix B.3.*

## 5.2 Expediting RL with Temporal-Causal Knowledge

In the *Generator Task* from Figure 1, if agent 1 unlocks the door  before fixing the pipe , it will not be able to return and fix the pipe later, because its path is blocked by a one-way ramp. Unfortunately, agent 1’s projected reward machine, illustrated in Figure 2a, does not capture this information. Due to this mismatch, during decentralized training episodes, agent 1 will tend to waste a large portion of time steps exploring trajectories which do not lead to a positive reward.

As discussed in Section 4, one can use high-level symbolic knowledge in the form of the TL-CD on Figure 3a to capture this information. While we first exploited this knowledge to check if the task specification for the *Generator Task* is decomposable into local task specifications, we will now use it to expedite decentralized training for agent 1. Note that the same TL-CD has been applied for both purposes: this is not always the case. In our second case study, which covers the *Laboratory Task*, we use a separate TL-CD to expedite decentralized training for two agents at once, but a different one to prove that the task specification passes the relaxed decomposition criterion.

Designing task specifications using reward machines is an error-prone and time-consuming task, and it is challenging to accommodate all possible scenarios and causal structures in advance. Moreover, manually updating reward machines can lead to unintended consequences, ultimately inducing a different optimal policy. Therefore, we investigate how to extend DQPRM to automatically incorporate temporal-causal knowledge about the environment, and help agents achieve a better balance exploration and exploitation, without adversely affecting performance in the original task.

To this end, we adapt the method proposed in Corazza et al. [3] to Event-based task-completion RMs. The method relies on finding *rejecting sink states* in the causal DFA  $\mathcal{C}$ , i.e., states  $q_{r.s.} \in Q^{\mathcal{C}} \setminus F^{\mathcal{C}}$  such that  $\delta_{\mathcal{C}}(q, e) = q$  for all  $e \in \Sigma_i$ . We implicitly consider causal DFAs to have at most one rejecting sink state  $q_{r.s.}$ , and that an accepting state is reachable from all other states. This can be achieved by minimization [9]. On Figure 3b, that is the state  $q_2$ . At its core, the method exploits the fact that once a label sequence  $\xi$  induces a transition into  $q_{r.s.}$ , there is no continuation  $\xi \cdot \xi'$  that will exit it, and are therefore, all such label sequences unattainable.

In order to identify explorations that will not be rewarded, the method computes  $\tilde{\mathcal{R}}_i$ , a modification of agent  $i$ ’s projected RM  $\mathcal{R}_i$  that induces the same optimal policy [3], but embeds the temporal-causal knowledge captured

by  $\mathcal{C}$ . The states of  $\tilde{\mathcal{R}}_i$  are elements of  $U^i \times Q^{\mathcal{C}}$  (the Cartesian product of the states of  $\mathcal{R}_i$  and the causal DFA of  $\mathcal{C}$ ), while the transition function  $\tilde{\delta}_i$  is defined as  $\tilde{\delta}_i((u, q), e) = (\delta_i(u, e), \delta_{\mathcal{C}}(q, e))$ . The reward function  $\tilde{\sigma}_i$  is defined as  $\tilde{\sigma}_i((u, q), (u', q')) = \sigma_i(u, u')$  if  $q \neq q_{\text{r.s.}}$ , and  $-1$  otherwise.<sup>6</sup>

Because only unattainable label sequences (such as *DGP* in the *Generator Task*) induce transitions into rejecting sink states, setting the reward to  $-1$  when reaching such states does not affect the optimal policy. The final step relies on computing the solution to the Bellman optimality equation over states of  $\tilde{\mathcal{R}}_i$ ,  $V^*(u, q) = \max_{e \in \Sigma_i, u' = \tilde{\delta}_i(u, e)} (\sigma((u, q), (u', q')) + V^*(u', q'))$ . In every step, the maximum possible return from the current episode is bounded from above by  $V^*(u, q)$  and  $0$  from below (by definition of task-completion RMs). If  $\tilde{\mathcal{R}}_i$  reaches a state  $(u, q)$  such that  $V^*(u, q) = 0$  during decentralized training, then all policies induce the same return thereon, and the learning can stop.

We summarize our method in Algorithm 2, which we call Causal DQPRM. The inputs for Causal DQPRM include a TL-CD  $\mathcal{C}$  which is used to ensure the task specification passes the relaxed decomposition criterion, along with a family of TL-CDs  $\mathcal{C}_1, \dots, \mathcal{C}_N$  (possibly a different one for each agent). In Algorithm 2,  $q_i$  ranges over the states of causal DFA  $\mathcal{C}_i$ , and  $(u_i, q_i)$  over the states of  $\tilde{\mathcal{R}}_i$ , the recomputed task specification for agent  $i$ . In our experiments, we use  $p = 0.3$ .

## 6 Case Studies

We performed three case studies, validating our approach in the *Generator Task*, described extensively throughout this paper, and two new domains: the *Laboratory Task*, and the *Buttons Task*. This section will detail the results on the *Generator Task* and the *Laboratory Task*, while the *Buttons Task* is described in Appendix C.

### 6.1 Case Study 1: Generator Task

Our first case study is an ablation of our approach on the *Generator Task*, with results shown in Figure 4. We compare the average steps needed for task completion per training step. The baseline we compare against is a centralized controller found with Q-learning, corresponding to the *No TL-CD* plot on Figure 4, *Centralized*.

Adding temporal-causal information improves team performance, even with centralized training. Decentralized training, enabled by our relaxed decomposition check, is significantly more efficient, taking an order of magnitude fewer steps to converge. In our two additional case studies, we perform the same analysis for the *Laboratory Task* and the *Buttons Task*.

**Algorithm 2:** Decentralized training with Causal DQPRM

---

**Input :** Team RM  $\mathcal{R}$ , local event sets  $\Sigma_1, \dots, \Sigma_N$ , local labeling functions  $L_1, \dots, L_N$ , TL-CDs  $\mathcal{C}, \mathcal{C}_1, \dots, \mathcal{C}_N$

**Output :** Policies  $\pi_1, \dots, \pi_N$

- 1 Project  $\mathcal{R}$  along local event sets  $\Sigma_i$  to obtain projected RMs  $\mathcal{R}_1, \dots, \mathcal{R}_N$ ;
- 2 **if**  $\mathcal{R} \parallel \mathcal{C} \cong (\parallel_{i=1}^N \mathcal{R}_i) \parallel \mathcal{C}$  **then**
- 3 | Reject: the relaxed decomposition criterion does not hold;
- 4 **end**
- 5 **for**  $i = 1$  **to**  $N$  **do**
- 6 | Compute  $\tilde{\mathcal{R}}_i$  via value iteration over  $\mathcal{R}_i \parallel \mathcal{C}_i$  ;
- 7 |  $s_i \leftarrow s_I^i, (u_i, q_i) \leftarrow (u_I^i, q_I^i)$ ,  $\text{steps}_i \leftarrow 0$ ;
- 8 **end**
- 9 Initialize( $\pi_1, \dots, \pi_N$ );
- 10 **for**  $t = 1$  **to** numEpisodes \* numSteps **do**
- 11 | **for**  $i = 1$  **to**  $N$  **do**
- 12 | | **if**  $\text{steps}_i > \text{numSteps}$  **or**  $V_i^*(u_i, q_i) = 0$  **then**
- 13 | | |  $s_i \leftarrow s_I^i, (u_i, q_i) \leftarrow (u_I^i, q_I^i)$ ,  $\text{steps}_i \leftarrow 0$ ;
- 14 | | **end**
- 15 | |  $a_i \leftarrow \text{Sample}(\pi_i(s_i, u_i))$ ;
- 16 | |  $s' \leftarrow \text{Step}_i(s_i, u_i, a_i), (u', q') \leftarrow (u_i, q_i)$ ,  $r_i \leftarrow 0$ ;
- 17 | | **foreach**  $e \in L_i(s_i, u_i, s')$  **do**
- 18 | | | **if**  $|I_e| = 1$  **or**  $\text{Rand}(0, 1) \leq p$  **then**
- 19 | | | |  $(u', q') \leftarrow \tilde{\delta}((u_i, q_i), e)$ ;
- 20 | | | |  $r_i \leftarrow r_i + \tilde{\sigma}((u_i, q_i), u')$ ;
- 21 | | | |  $(u_i, q_i) \leftarrow (u', q')$ ;
- 22 | | | **end**
- 23 | | **end**
- 24 | |  $\text{PolicyUpdate}(\pi_i, s_i, u_i, a_i, s', r_i)$ ;
- 25 | |  $s_i \leftarrow s'$ ,  $\text{steps}_i \leftarrow \text{steps}_i + 1$ ;
- 26 | **end**
- 27 **end**
- 28 **return**  $(\pi_1, \dots, \pi_N)$ ;

---

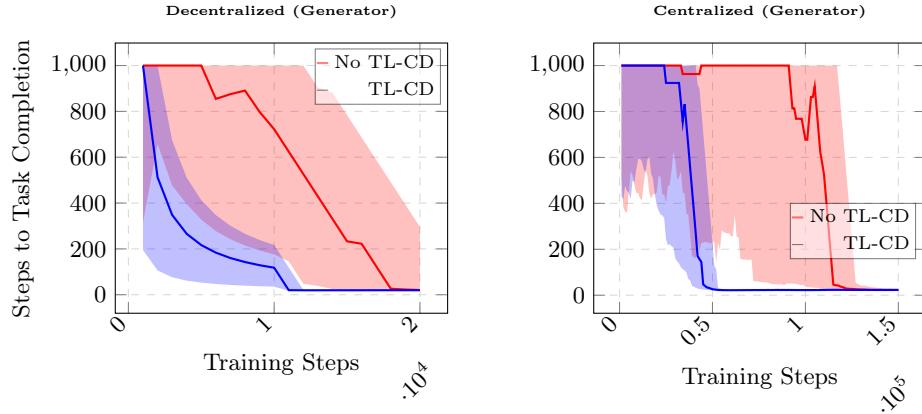
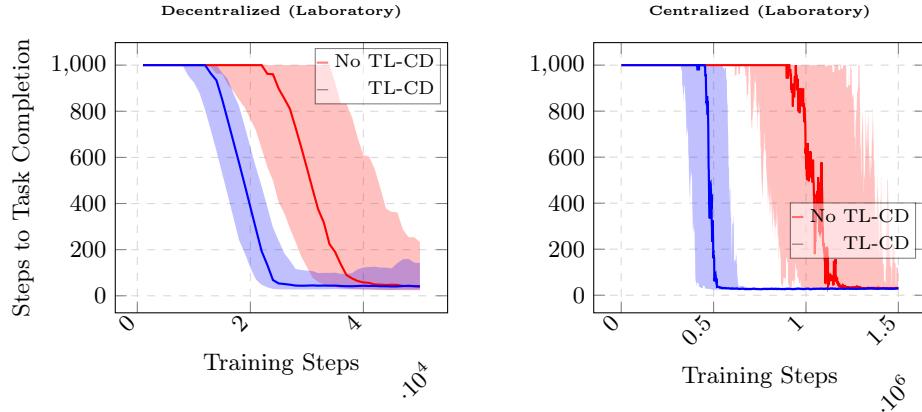
## 6.2 Case Study 2: Laboratory Task

In the *Laboratory Task*, two agents are tasked with aiding with an accident that occurred in a laboratory. There are two possible types of accidents that may have occurred: a fire, represented by , or a radioactive spill, represented by . Agent 1 is equipped with heat sensors, but not with radiation sensors, vice-versa for agent 2.

The two accident types are mutually exclusive. Once the agents enter the conveyor belt, represented by , which leads them to the laboratory, either agent 1 will observe , or agent 2 will observe  (with 50% probability each). Once

---

<sup>6</sup> To justify this we note that, for task-completion RMs,  $-1$  satisfies the reward bounds computed in Corazza et al. [3].

Fig. 4: *Generator Task* study. Aggregated results from 50 independent runs.Fig. 5: *Laboratory Task* study. Aggregated results from 50 independent runs.

inside, the agents must, depending on the type of the accident, converge to a tool which will provide aid (----

We illustrate the environment for the *Laboratory Task* in Figure 6a, and the team reward machine which provides the task specification in Figure 6b. As shown by the results in Figure 5, Causal DQPRM is able to solve this task, and leverages temporal-causal information to significantly increase sample efficiency compared to a centralized controller.

On Figure 7, we illustrate the local reward machines of agents 1 and 2 in the *Laboratory Task*.

Our experiments were conducted on a single machine with AMD Ryzen 7 5825U and 6.9G of RAM, except for the centralized case of the Buttons task, where we used a machine with AMD EPYC 7742 64-Core Processor and 515G of RAM, due to the extremely large state space.

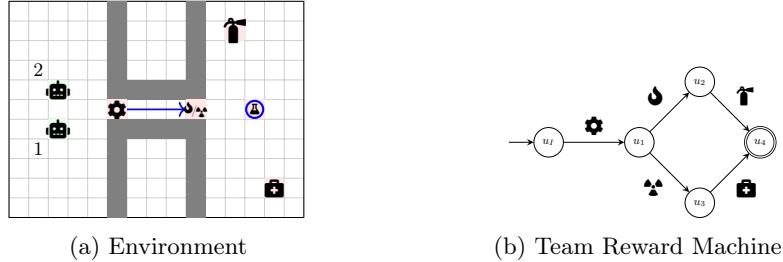
Fig. 6: *Laboratory Task*

Fig. 7: Projections of the RM from Figure 6b along local event sets of agents 1 (Left) and 2 (Right).

## 7 Conclusion

We introduced a framework for integrating temporal-causal knowledge, formalized via TL-CDs, into decentralized multi-agent reinforcement learning, enabling both relaxed task decomposition and accelerated policy learning. Experimentally validated across three case studies, our method improves the applicability of decentralized training guarantees while achieving higher success rates and sample efficiency compared to baseline approaches. By automating the incorporation of expert knowledge into task specifications and learning processes, this work enables scalable solutions to complex, temporally structured multi-agent tasks.

**Acknowledgments.** This work was supported in part by the National Science Foundation (NSF) under Grants CNS 2304863 and CNS 2339774, and in part by the Office of Naval Research (ONR) under Grant N00014-23-1-2505.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Almeida, M., Moreira, N., Reis, R.: Testing the equivalence of regular languages. In: Dassow, J., Pighizzini, G., Truthe, B. (eds.) Proceedings Eleventh International Workshop on Descriptive Complexity of Formal Systems, DCFS 2009, Magdeburg, Germany, July 6-9, 2009. EPTCS, vol. 3, pp. 47–57 (2009). <https://doi.org/10.4204/EPTCS.3.4>, <https://doi.org/10.4204/EPTCS.3.4>

2. Azran, G., Danesh, M.H., Albrecht, S.V., Keren, S.: Contextual pre-planning on reward machine abstractions for enhanced transfer in deep reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence **38**(10), 10953–10961 (Mar 2024). <https://doi.org/10.1609/aaai.v38i10.28970>, <https://ojs.aaai.org/index.php/AAAI/article/view/28970>
3. Corazza, J., Aria, H.P., Neider, D., Xu, Z.: Expediting reinforcement learning by incorporating knowledge about temporal causality in the environment. In: Locatello, F., Didelez, V. (eds.) Proceedings of the Third Conference on Causal Learning and Reasoning. Proceedings of Machine Learning Research, vol. 236, pp. 643–664. PMLR (01–03 Apr 2024), <https://proceedings.mlr.press/v236/corazza24a.html>
4. Dohmen, T., Topper, N., Atia, G., Beckus, A., Trivedi, A., Velasquez, A.: Inferring probabilistic reward machines from non-markovian reward processes for reinforcement learning (2022)
5. Icarte, R.T., Klassen, T.Q., Valenzano, R.A., McIlraith, S.A.: Reward machines: Exploiting reward function structure in reinforcement learning. CoRR **abs/2010.03950** (2020), <https://arxiv.org/abs/2010.03950>
6. Kazemi, M., Perez, M., Somenzi, F., Soudjani, S., Trivedi, A., Velasquez, A.: Assume-guarantee reinforcement learning (2023)
7. Neary, C., Xu, Z., Wu, B., Topcu, U.: Reward machines for cooperative multi-agent reinforcement learning. CoRR **abs/2007.01962** (2020), <https://arxiv.org/abs/2007.01962>
8. Paliwal, Y., Roy, R., Gaglione, J.R., Baharisangari, N., Neider, D., Duan, X., Topcu, U., Xu, Z.: Reinforcement learning with temporal-logic-based causal diagrams (2023)
9. Sipser, M.: Introduction to the Theory of Computation. Course Technology, Boston, MA, third edn. (2013)
10. Wong, K.C.: On the complexity of projections of discrete-event systems. Discrete Event Dynamic Systems (1998), <https://api.semanticscholar.org/CorpusID:17637223>
11. Xu, Z., Gavran, I., Ahmad, Y., Majumdar, R., Neider, D., Topcu, U., Wu, B.: Joint inference of reward machines and policies for reinforcement learning. Proceedings of the International Conference on Automated Planning and Scheduling **30**, 590–598 (Jun 2020). <https://doi.org/10.1609/icaps.v30i1.6756>, <http://dx.doi.org/10.1609/icaps.v30i1.6756>

## A Parallel Composition and Bisimilarity of Reward Machines

In this section, we introduce the formal definitions of parallel composition and bisimilarity for reward machines.

**Definition 5 (Parallel Composition of Reward Machines).** Let  $\mathcal{R}_1 = \langle U_1, u_I^1, \Sigma_1, \delta_1, \sigma_1, F_1 \rangle$  and  $\mathcal{R}_2 = \langle U_2, u_I^2, \Sigma_2, \delta_2, \sigma_2, F_2 \rangle$  be two reward machines. The parallel composition of  $\mathcal{R}_1$  and  $\mathcal{R}_2$  is a new reward machine  $\mathcal{P} = \langle U, u_I, \Sigma, \delta, \sigma, F \rangle$  where the

- set of states is  $U = U_1 \times U_2$ ,
- initial states is  $u_I = (u_I^1, u_I^2)$ ,
- input alphabet is  $\Sigma = \Sigma_1 \cup \Sigma_2$
- transition function is defined as

$$\delta((u_1, u_2), e) = \begin{cases} (\delta_1(u_1, e), \delta_2(u_2, e)) & \text{if } \delta_1(u_1, e) \text{ and} \\ & \delta_2(u_2, e) \text{ are defined} \\ (\delta_1(u_1, e), u_2) & \text{if only } \delta_1(u_1, e) \text{ is defined} \\ (u_1, \delta_2(u_2, e)) & \text{if only } \delta_2(u_2, e) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

- set of terminal states is  $F = F_1 \times F_2$ ; and
- output function  $\sigma : U \times U \rightarrow \mathbb{R}$  is defined as  $\sigma(u, u') = 1$  if  $u' \in F$  and  $u \notin F$  and 0 otherwise.

**Definition 6 (Bisimilarity of Reward Machines).** Let  $\mathcal{R}_1 = \langle U_1, u_I^1, \Sigma, \delta_1, \sigma_1, F_1 \rangle$  and  $\mathcal{R}_2 = \langle U_2, u_I^2, \Sigma, \delta_2, \sigma_2, F_2 \rangle$  be two reward machines with the same event set  $\Sigma$ .  $\mathcal{R}_1$  and  $\mathcal{R}_2$  are bisimilar ( $\mathcal{R}_1 \cong \mathcal{R}_2$ ) if there exists a relation  $\mathcal{R} \subseteq U_1 \times U_2$  such that

1.  $(u_I^1, u_I^2) \in \mathcal{R}$ .
2. For every  $(u_1, u_2) \in \mathcal{R}$  and  $e \in \Sigma$ :
  - (**Accepting**)  $u_1 \in F_1$  if and only if  $u_2 \in F_2$ .
  - (**Forth**) if  $\delta_1(u_1, e) = u'_1$ , then there exists  $u'_2 \in U_2$  such that  $\delta_2(u_2, e) = u'_2$  and  $(u'_1, u'_2) \in \mathcal{R}$ .
  - (**Back**) if  $\delta_2(u_2, e) = u'_2$ , then there exists  $u'_1 \in U_1$  such that  $\delta_1(u_1, e) = u'_1$  and  $(u'_1, u'_2) \in \mathcal{R}$ .

## B Proofs

### B.1 Proof of Theorem 3 (Relaxed Decomposition Criterion)

*Proof.* Let  $\mathcal{R}$  be the team task RM, and  $\mathcal{R}_1, \dots, \mathcal{R}_N$  be the projections of  $\mathcal{R}$  onto the local event sets  $\Sigma_1, \dots, \Sigma_N$ , respectively. Let  $\mathcal{C}$  be a TL-CD that holds

for  $\mathcal{M}$ , with equivalent causal DFA  $\mathsf{C}$ , such that  $\mathcal{R} \parallel \mathsf{C} \cong (\|_{i=1}^N \mathcal{R}_i) \parallel \mathsf{C}$ , and let  $\xi \in \Sigma^*$  be an  $\mathcal{M}$ -attainable event sequence.

By assumption, we have that  $\xi \models \varphi_{\Sigma}^{\mathsf{C}}$ . If  $\mathcal{R}(\xi) = 0$ , then  $\mathcal{R} \parallel \mathsf{C} = 0$ , by definition of parallel composition. Due to  $\mathcal{R} \parallel \mathsf{C} \cong (\|_{i=1}^N \mathcal{R}_i) \parallel \mathsf{C}$ , we also have  $(\|_{i=1}^N \mathcal{R}_i) \parallel \mathsf{C} = 0$ . Here, we used the fact that the pair of reward machines  $\mathcal{R} \parallel \mathsf{C}$  and  $(\|_{i=1}^N \mathcal{R}_i) \parallel \mathsf{C}$  are bisimilar, and pass the *strict* decomposition criterion 1. Because  $\mathsf{C}(\xi) = 1$  (i.e.,  $\delta_{\mathsf{C}}(q_I^{\mathsf{C}}, \xi) \in F^{\mathsf{C}}$ ), and the definition of parallel composition, we have  $(\|_{i=1}^N \mathcal{R}_i)(\xi) = 0$ . Again by the definition of parallel composition, we have that  $\exists i \in \{1, \dots, N\} : \mathcal{R}_i(P_i(\xi)) = 0$ . The converse direction proceeds analogously.  $\square$

## B.2 Proof of Theorem 4 (Criterion Compatibility)

*Proof.* The proof is direct. Assume  $\rho \subseteq U^{\mathcal{R}} \times U^{\mathcal{P}}$  is a bisimulation between  $\mathcal{R}$  and  $\mathcal{P}$ .

Let  $Q^{\mathsf{C}}$  be the set of states of the causal DFA  $\mathsf{C}$ . We define the relation  $\tilde{\rho} \subseteq U_{\mathcal{R} \parallel \mathsf{C}} \times U_{\mathcal{P} \parallel \mathsf{C}}$  via Formula 1, for all states  $(u^{\mathcal{R}}, q_1) \in U_{\mathcal{R} \parallel \mathsf{C}}$  and  $(u^{\mathcal{P}}, q_2) \in U_{\mathcal{P} \parallel \mathsf{C}}$ .

$$((u^{\mathcal{R}}, q_1), (u^{\mathcal{P}}, q_2)) \in \tilde{\rho} \Leftrightarrow (u^{\mathcal{R}}, u^{\mathcal{P}}) \in \rho \wedge q_1 = q_2 \quad (1)$$

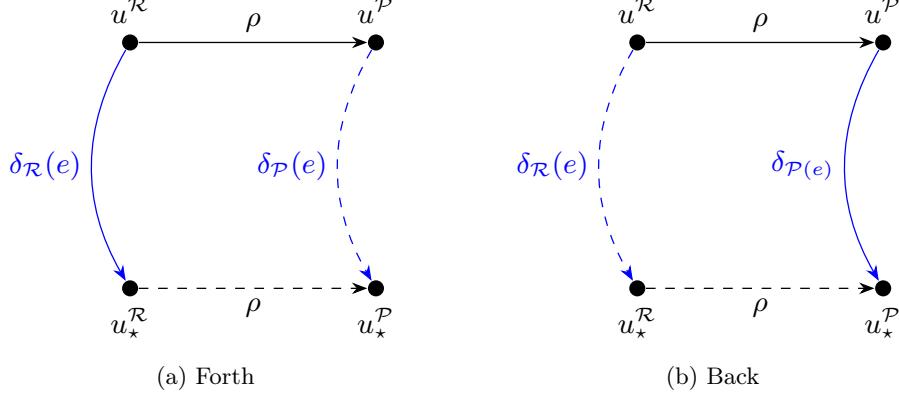
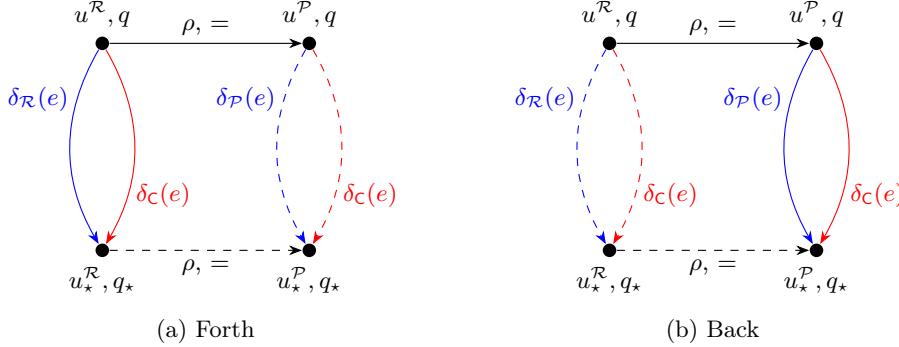
By checking the bisimulation conditions, we will show that  $\tilde{\rho}$  is a bisimulation between  $\mathcal{R} \parallel \mathsf{C}$  and  $\mathcal{P} \parallel \mathsf{C}$ .

1.  $((u_I^{\mathcal{R}}, q_I), (u_I^{\mathcal{P}}, q_I)) \in \tilde{\rho}$  because  $(u_I^{\mathcal{R}}, u_I^{\mathcal{P}}) \in \rho$ .
2. Let  $((u^{\mathcal{R}}, q_1), (u^{\mathcal{P}}, q_2)) \in \tilde{\rho}$ , and  $e \in \Sigma$  some event. Then  $(u^{\mathcal{R}}, u^{\mathcal{P}}) \in \rho$  and  $q_1 = q_2 = q \in Q^{\mathsf{C}}$ . Therefore, we have the following.
  - **(Accepting)**  $(u^{\mathcal{R}}, q) \in F^{\mathcal{R} \parallel \mathsf{C}}$  if and only if  $(u^{\mathcal{P}}, q) \in F^{\mathcal{P} \parallel \mathsf{C}}$ , because  $(u^{\mathcal{R}}, q) \in F^{\mathcal{R} \parallel \mathsf{C}} \Leftrightarrow u^{\mathcal{R}} \in F^{\mathcal{R}} \wedge q \in F^{\mathsf{C}} \Leftrightarrow u^{\mathcal{P}} \in F^{\mathcal{P}} \wedge q \in F^{\mathsf{C}} \Leftrightarrow (u^{\mathcal{P}}, q) \in F^{\mathcal{P} \parallel \mathsf{C}}$ .
  - **(Forth)** if  $\delta_{\mathcal{R} \parallel \mathsf{C}}((u^{\mathcal{R}}, q), e) = (u_{\star}^{\mathcal{R}}, q_{\star})$ , then there exists  $u_{\star}^{\mathcal{P}}$  such that  $\delta_{\mathcal{P} \parallel \mathsf{C}}((u^{\mathcal{P}}, q), e) = (u_{\star}^{\mathcal{P}}, q_{\star})$  and  $((u_{\star}^{\mathcal{R}}, q_{\star}), (u_{\star}^{\mathcal{P}}, q_{\star})) \in \tilde{\rho}$ .
  - **(Back)** if  $\delta_{\mathcal{P} \parallel \mathsf{C}}((u^{\mathcal{P}}, q), e) = (u_{\star}^{\mathcal{P}}, q_{\star})$ , then there exists  $u_{\star}^{\mathcal{R}}$  such that  $\delta_{\mathcal{R} \parallel \mathsf{C}}((u^{\mathcal{R}}, q), e) = (u_{\star}^{\mathcal{R}}, q_{\star})$  and  $((u_{\star}^{\mathcal{R}}, q_{\star}), (u_{\star}^{\mathcal{P}}, q_{\star})) \in \tilde{\rho}$ .

One obtains  $u_{\star}^{\mathcal{P}}$  ( $u_{\star}^{\mathcal{R}}$ ) from the bisimulation condition on  $\mathcal{P}$  and  $\mathcal{R}$ . Then,  $((u_{\star}^{\mathcal{P}}, q_{\star}), (u_{\star}^{\mathcal{R}}, q_{\star})) \in \tilde{\rho}$  by definition of  $\tilde{\rho}$ . The diagrams in Figure 8, Figure 9, and Figure 10 illustrate the back and forth bisimulation conditions.  $\square$

## B.3 Proof of Theorem 5 (Relaxed Decomposition Viability)

*Proof.* Let  $\mathcal{M} = \langle S, s_I, A, p, \gamma, \mathcal{R}, L \rangle$  be the team task RM-MDP, with local event sets  $\Sigma_1, \dots, \Sigma_N$  and decomposable label function  $L, L_1, \dots, L_N$  the local labeling functions of agents  $1, \dots, N$ , and  $\mathcal{C}$  the TL-CD which holds for  $\mathcal{M}$ . Assume that

Fig. 8: Visualizing  $\mathcal{R} \cong \mathcal{P}$ .Fig. 9: Visualizing  $\mathcal{R} \cong \mathcal{P}$  alongside transitions from  $\mathcal{C}$ .

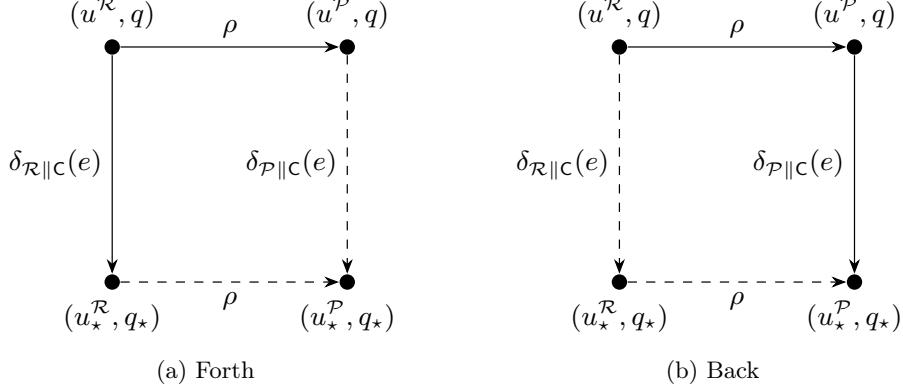
the relaxed decomposition criterion holds, and that agents synchronize on shared events.

We use  $\langle L_i(s_0^i u_0^i \cdots s_k^i u_k^i) \mid \text{Sync}^i \rangle$  to denote the synchronized local labeling function output.  $\text{Sync}^i$  is a binary vector, and  $\text{Sync}_t^i = 1$  iff. agent  $i$  received a synchronization signal in time-step  $t$ .

For the first claim we need to show that for all team trajectories  $s_0 u_0 \cdots s_k u_k$  and local trajectories  $\{s_0^i u_0^i \cdots s_k^i u_k^i\}_{i=1}^N$ , we have  $\mathcal{R}(L(s_0 u_0 \cdots s_k u_k)) = 1$  if and only if  $\mathcal{R}_i(\langle L_i(s_0^i u_0^i \cdots s_k^i u_k^i) \mid \text{Sync}^i \rangle) = 1$  for all  $i = 1, \dots, N$ .

We first note that  $s_0 u_0 \cdots s_k u_k$  is sampled from  $\mathcal{M}$ , which means that  $L(s_0 u_0 \cdots s_k u_k) \models \varphi_{\Sigma}^{\mathcal{C}}$ . Then the result follows from Theorem 3 if we show that  $P_i(L(s_0 u_0 \cdots s_k u_k)) = \langle L_i(s_0^i u_0^i \cdots s_k^i u_k^i) \mid \text{Sync}^i \rangle$  (for every  $i = 1, \dots, N$ ). Let  $\ell_t$  denote the output of  $L$  at time-step  $t = 0, \dots, k$ ,  $\ell_t^i$  the output of  $L_i$ , and  $\tilde{\ell}_t^i$  the synchronized output of  $L_i$ .

We proceed by induction over time-steps  $t$ .

Fig. 10: Visualizing  $\mathcal{R} \parallel \mathcal{C} \cong \mathcal{P} \parallel \mathcal{C}$ .

**Base case:**  $t = 0$ . In that case, by definition,  $\ell_0 = \ell_0^i = \tilde{\ell}_0^i = \emptyset$ . Additionally, by definition,  $u_0 = u_I \in u_I^i = u_0^i$  for all  $i = 1, \dots, N$  (the initial state  $u_0$  of  $\mathcal{R}$  is matched by the initial states  $u_0^i$  of all projections  $\mathcal{R}_i$ ).

**Assumption:** Let  $t \in \mathbb{N}$ ,  $0 < t < k$  such that for all  $i = 1, \dots, N$  it holds that  $\ell_t \cap \Sigma_i = \tilde{\ell}_t^i$ , and  $u_t \in u_t^i$ .

**Inductive step:** By the inductive hypothesis  $u_t \in u_t^i$ . By the decomposability of  $L$ , the output of  $L_i(s_t^i, u_t^i, s_{t+1}^i) = \ell_{t+1}^i$  is well-defined. There are three cases to consider: (1)  $\ell_{t+1}^i = \ell_{t+1} \cap \Sigma_i = \{e\}$  and  $|I_e| > 1$ ; (2)  $\ell_{t+1}^i = \ell_{t+1} \cap \Sigma_i = \{e\}$  and  $|I_e| = 1$ ; and (3)  $\ell_{t+1}^i = \emptyset$ .

In cases (2) and (3), we immediately have  $\ell_{t+1}^i = \ell_{t+1} \cap \Sigma_i = \tilde{\ell}_{t+1}^i$  (no synchronization necessary), and by definition of parallel projections and the inductive hypothesis,  $\delta(u_t, \ell_{t+1}) = u_{t+1} \in u_{t+1}^i = \delta^i(u_t^i, \tilde{\ell}_{t+1}^i)$ .

In case (1), for all  $i \in I_e$  we have  $\text{Sync}_{t+1}^i = 1$  if  $e \in \ell_{t+1}$ , or  $\text{Sync}_{t+1}^i = 0$  otherwise. If  $\text{Sync}_{t+1}^i = 1$ , then  $\tilde{\ell}_{t+1}^i = \{e\}$ , otherwise  $\tilde{\ell}_{t+1}^i = \emptyset$ . The rest of case (1) follows analogously to cases (2) and (3), and we may conclude that  $u_{t+1} \in \delta^i(u_t^i, \tilde{\ell}_{t+1}^i)$  for all  $i = 1, \dots, N$  in all cases, i.e. the  $\sim_i$ -classes of the projected RMAs match the state of the team RM in all steps.

That, in turn, implies  $P_i(L(s_0 u_0 \cdots s_k u_k)) = \langle L_i(s_0^i u_0^i \cdots s_k^i u_k^i) \mid \text{Sync}^i \rangle$  for all  $i = 1, \dots, N$ , and the first claim is proven via Theorem 3 and  $L(s_0 u_0 \cdots s_k u_k) \models \varphi_{\Sigma}^C$ .

The second claim is that  $\max\{0, V_1^\pi(s_I) + \cdots + V_N^\pi(s_I) - (N-1)\} \leq V^\pi(s_I) \leq \min\{V_1^\pi(s_I), \dots, V^\pi(s_I)\}$ , where  $V^\pi(s_I)$  denotes the team success probability, and  $V_i^\pi(s_I)$  the success probability of agent  $i$ . We use the notation  $V^\pi$  to evoke the correspondence between the success probability and the value function in non-discounted task-completion RM-MDPs ( $\gamma = 1$ ).

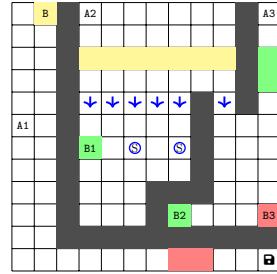
By the previous result, for any sequence of team states  $s_0 u_0 \cdots s_k u_k$ , we have  $\mathcal{R}(L(s_0 u_0 \cdots s_k u_k)) = 1$  if and only if  $\mathcal{R}_i(\langle L_i(s_0^i u_0^i \cdots s_k^i u_k^i) \mid \text{Sync}^i \rangle) = 1$  for all  $i = 1, \dots, N$ . This implies that the probability of completing the task captured

by  $\mathcal{R}$  under policy  $\pi$  is equal to the probability of simultaneously completing all tasks captured by  $\mathcal{R}_i$  (when  $\pi = (\pi_1, \dots, \pi_N)$ ). Then the second claim follows by applying the Fréchet conjunction inequality to the success probabilities of the agents.  $\square$

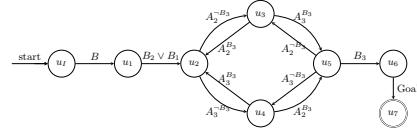
### C Case Study 3: Buttons Task

In our third case study, we perform the same analysis for a 3-agent problem we call the *Cooperative Buttons* Task. In this environment shown in Figure 11a, events  $B$ ,  $B_1$ ,  $B_2$ , and  $B_3$  correspond to yellow, green, and red buttons respectively. In this shared environment, there are three agents - A1, A2, and A3 - who have to work together to achieve the objective of allowing A1 to reach the goal location known as Goal. However, the path to the goal is blocked by three colored regions - red, yellow, and green - which correspond to the paths of agents A1, A2, and A3 respectively. To cross these colored regions, the corresponding colored button must be pressed first. The yellow and green buttons can be pressed by an individual agent, but the red button requires two agents to simultaneously occupy the button's location before it gets activated.

In order to complete the task, the agents have to follow a specific sequence of events. First, A1 should push the yellow button, which will allow A2 to proceed to the green button. Pressing the green button is necessary for A3 to join A2 in pressing the red button, which will finally allow A1 to cross the red region and reach the goal location. The sequence of events is encoded in the reward machine depicted in Figure 11b.



(a)  $B$ ,  $B_1$ ,  $B_2$ , and  $B_3$  are buttons, and  $\blacksquare$  is the goal. One-way doors are represented by the downwards arrow. States labeled with  $S$  are signal lights which indicate that the agent is in the region that cannot go to the red button.



(b) Reward machine encoding the cooperative buttons task. Output is 1 on transitions into accepting states, and 0 otherwise.

Fig. 11: Environment (left) and RM (right) for the *cooperative buttons* task. One-way doors block agent 1 to go to the red button and push the button.

The results for this case study are shown in Figure 12.

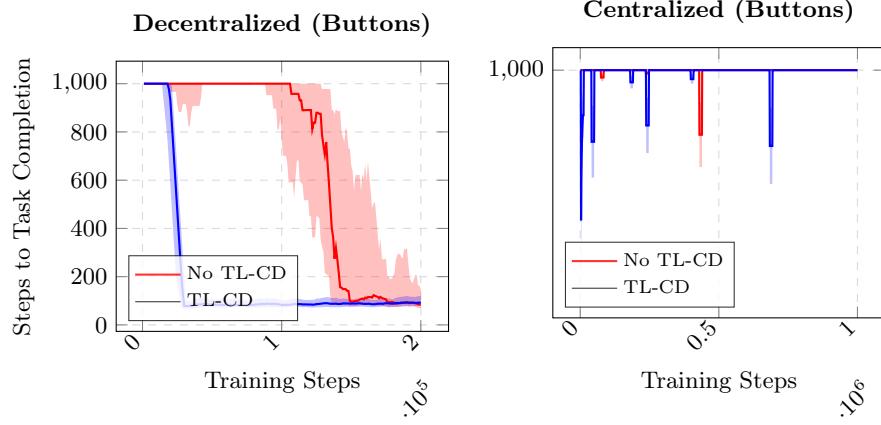


Fig. 12: Buttons Task Comparisons. Results of 10 independent runs (2 in the case of the Centralized algorithm, which does not converge in  $10^6$  steps).

Figures 13a, 13b, and 13c show the results of projecting the task RM from Figure 11b onto the local event sets of  $\Sigma_1$ ,  $\Sigma_2$ , and  $\Sigma_3$ , respectively.

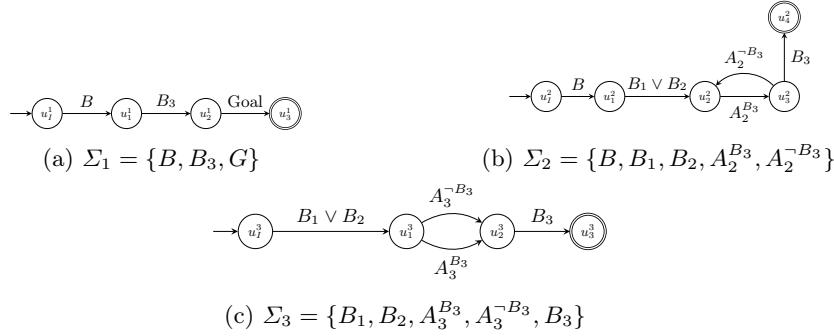


Fig. 13: Projections of the team task RM from Figure 11b along local event sets of agents 1, 2, and 3.

In the cooperative button task, states labeled with S indicate that the agent is in a region from which it cannot proceed to the red button and, consequently, cannot reach the goal. These labels, called 'Signal lights', serve as indicators of non-goal-achievable states, providing critical information about the agent's position within the environment. This knowledge is encoded in Figure 14, which represents the TL-CD for the cooperative button task. The TL-CD visually outlines the causal pathways and dependencies between states, highlighting the impact of signal lights on the agent's progress. Complementing this diagram, the

related Causal DFA formalizes these causal relationships and transitions, aiding in understanding and predicting the agent's behavior within the task.

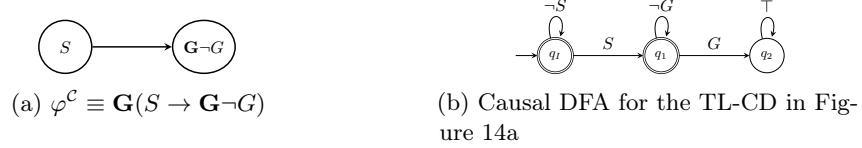


Fig. 14: TL-CD for the Cooperative Buttons Task (Left) and respective Causal DFA (Right).